



PAPER • **OPEN ACCESS**

Fermion-to-qubit mappings with varying resource requirements for quantum simulation

To cite this article: Mark Steudtner and Stephanie Wehner 2018 *New J. Phys.* **20** 063010

View the [article online](#) for updates and enhancements.

Related content

- [Optimizing qubit resources for quantum chemistry simulations in second quantization on a quantum computer](#)
Nikolaj Moll, Andreas Fuhrer, Peter Staar et al.
- [Quantum optimization using variational algorithms on near-term quantum devices](#)
Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop et al.
- [Frustration-free Hamiltonians supporting Majorana zero edge modes](#)
Sanja Jevtic and Ryan Barnett



IOP | ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the collection - download the first chapter of every title for free.



OPEN ACCESS

RECEIVED

4 January 2018

REVISED

6 April 2018

ACCEPTED FOR PUBLICATION

16 May 2018

PUBLISHED

7 June 2018

Original content from this work may be used under the terms of the [Creative Commons Attribution 3.0 licence](#).

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.



PAPER

Fermion-to-qubit mappings with varying resource requirements for quantum simulation

Mark Steudtner^{1,2} and Stephanie Wehner²¹ Instituut-Lorentz, Universiteit Leiden, PO Box 9506, 2300 RA Leiden, The Netherlands² QuTech, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The NetherlandsE-mail: steudtner@lorentz.leidenuniv.nl**Keywords:** quantum simulation, quantum algorithms, quantum chemistry

Abstract

The mapping of fermionic states onto qubit states, as well as the mapping of fermionic Hamiltonian into quantum gates enables us to simulate electronic systems with a quantum computer. Benefiting the understanding of many-body systems in chemistry and physics, quantum simulation is one of the great promises of the coming age of quantum computers. Interestingly, the minimal requirement of qubits for simulating Fermions seems to be agnostic of the actual number of particles as well as other symmetries. This leads to qubit requirements that are well above the minimal requirements as suggested by combinatorial considerations. In this work, we develop methods that allow us to trade-off qubit requirements against the complexity of the resulting quantum circuit. We first show that any classical code used to map the state of a fermionic Fock space to qubits gives rise to a mapping of fermionic models to quantum gates. As an illustrative example, we present a mapping based on a nonlinear classical error correcting code, which leads to significant qubit savings albeit at the expense of additional quantum gates. We proceed to use this framework to present a number of simpler mappings that lead to qubit savings with a more modest increase in gate difficulty. We discuss the role of symmetries such as particle conservation, and savings that could be obtained if an experimental platform could easily realize multi-controlled gates.

1. Introduction

Simulating quantum systems on a quantum computer is one of the most promising applications of small scale quantum computers [1]. Significant efforts have gone into the theoretical development of simulation algorithms [2–6], and their experimental demonstrations [7–12]. Resource estimates [13–15], such as for example for FeMoCo, a model for the nitrogenase enzyme, indicate that simulations of relevant chemical systems may be achieved with relatively modest quantum computing resources [16] in comparison to many standard quantum algorithms [17, 18].

One essential component in realizing simulations of fermionic models on quantum computers is the representation of such models in terms of qubits and quantum gates. Following initial simulation schemes for fermions hopping on a lattice [19], more recent proposals used the Jordan–Wigner [20] transform [3, 7, 21, 22], the Verstraete–Cirac mapping [23], or the Bravyi–Kitaev transform [2] to find a suitable representation. Specifically, the task of all such representations is two-fold. First, we seek a mapping from states in the fermionic Fock space of N sites to the space of n qubits. The fermionic Fock space is spanned by 2^N basis vectors $|\nu_1, \dots, \nu_N\rangle$ where $\nu_j \in \{0, 1\}$ indicates the presence ($\nu_j = 1$) or absence ($\nu_j = 0$) of a spinless fermionic particle at orbital j ³. Such a mapping $\mathbf{e}: \mathbb{Z}_2^{\otimes N} \rightarrow \mathbb{Z}_2^{\otimes n}$ is also called an *encoding* [24]. An example of such an encoding is the trivial one in which $n = N$ and qubits are used to represent the binary string $\boldsymbol{\nu} = (\nu_1, \dots, \nu_N)^T$. That is

³ We slightly abuse the nomenclature of quantum chemistry and molecular physics in merging spatial and spin quantum numbers into one index j , and use it as a label for what we call now the j th orbital.

$$|\omega\rangle = |\mathbf{e}(\nu)\rangle = \bigotimes_{j=1}^n |\omega_j\rangle, \quad (1)$$

where $\omega_j = \nu_j$ in the standard basis $\{|0\rangle, |1\rangle\}$.

Second, we need a way to simulate the dynamics of Fermions on these N orbitals. These dynamics can be modeled entirely in terms of the annihilation and creation operators c_j and c_j^\dagger that satisfy the anticommutation relations

$$[c_i, c_j]_+ = 0, \quad [c_i^\dagger, c_j^\dagger]_+ = 0, \quad [c_i, c_j^\dagger]_+ = \delta_{ij}, \quad (2)$$

with $[A, B]_+ = AB + BA$. Following these relations, the operators act on the fermionic Fock space as

$$c_{i_m}^\dagger c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_m}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle = 0, \quad (3)$$

$$c_{i_m} c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_m}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle = 0, \quad (4)$$

$$c_{i_m} c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_m}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle = (-1)^{m-1} c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle, \quad (5)$$

$$c_{i_m}^\dagger c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_m}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle = (-1)^{m-1} c_{i_1}^\dagger \dots c_{i_{m-1}}^\dagger c_{i_m}^\dagger c_{i_{m+1}}^\dagger \dots c_{i_M}^\dagger |\Theta\rangle, \quad (6)$$

where $|\Theta\rangle$ is the fermionic vacuum and $\{i_1, \dots, i_M\} \subseteq \{1, \dots, N\}$. Mappings of the operators c_j to qubits typically use the Pauli matrices X, Z , and Y acting on one qubit, characterized by their anticommutation relations $[P_i, P_j]_+ = 2\delta_{ij}\mathbb{I}$ for all $P_i \in \mathcal{P} = \{X, Z, Y\}$. An example of such a mapping is the Jordan–Wigner transform [20] given by

$$c_j \triangleq Z^{\otimes j-1} \otimes \sigma^- \otimes \mathbb{I}^{\otimes n-j}, \quad (7)$$

$$c_j^\dagger \triangleq Z^{\otimes j-1} \otimes \sigma^+ \otimes \mathbb{I}^{\otimes n-j}, \quad (8)$$

where

$$\sigma^- = |0\rangle\langle 1| = \frac{1}{2}(X + iY), \quad (9)$$

$$\sigma^+ = |1\rangle\langle 0| = \frac{1}{2}(X - iY). \quad (10)$$

It is easily verified that together with the trivial encoding (1) this transformation satisfies the desired properties (2)–(6) and can hence be used to represent fermionic models with qubit systems.

In order to assess the suitability of an encoding scheme for the simulation of fermionic models on a quantum computer, a number of parameters are of interest. The first is the total number of qubits n needed in the simulation. Second, we may care about the gate size of the operators c_j and c_j^\dagger when mapped to qubits. In its simplest form, this problem concerns the total number of qubits on which these operators do not act trivially, that is, the number of qubits L , on which an operator acts as $P_j \in \mathcal{P}$ instead of the identity \mathbb{I} , sometimes called the Pauli length. Different transformations can lead to dramatically different performance with respect to these parameters. For both the Jordan–Wigner as well as the Bravyi–Kitaev transform $n = N$, but we have $L = O(n)$ for the first, while $L = O(\log n)$ for the second. We remark that in experimental implementations we typically do not only care about the absolute number L , but rather the specific gate size and individual difficulty of the qubit gates each of which may be easier or harder to realize in a specific experimental architecture. For error-corrected quantum simulation, the cost in T-gates is as important to optimize as the circuit depth [25], and quantum devices with restricted connectivity even require mappings tailored to them [26, 27]. Finally, we remark that instead of looking for a mapping for individual operators $c_j^{(\dagger)}$ we may instead opt to map pairs (or higher order terms) of such operators at once, or even look to represent sums of such operators.

1.1. Results

Here, we propose a general family of mappings of fermionic models to qubit systems and quantum gates that allow us to trade-off the necessary number of qubits n against the difficulty of implementation as parametrized by L , or more complicated quantum gates such as CPHASE. Ideally, one would of course like both the number of qubits, as well as the gate size to be small. We show that our mappings can lead to significant savings in qubits for a variety of examples (see table 1) as compared to the Jordan–Wigner transform for instance, at the expense of greater complexity in realizing the required gates. The latter may lead to an increased time required for the simulation depending on which gates are easy to realize in a particular quantum computing architecture.

Table 1. Overview of mappings presented in this paper, listed by the complexity of their code functions, their qubit savings, qubit requirements (n), properties of the resulting gates and first appearance. Mappings can be compared with respect to the size of plain words (N) and their targeted Hamming weight K . We also refer to different methods that are not listed, as they do not rely on codes in any way [30, 31].

Mapping	En-/decoding type	Qubits saved	$n(N, K)$	Resulting gates	Origin
Jordan–Wigner Parity transform	Linear/linear	None	N	Length- $O(n)$ Pauli strings	[20, 24]
Bravyi–Kitaev transform	Linear/linear	None	N	Length- $O(\log n)$ Pauli strings	[2]
Checksum codes	Linear/ affine linear	$O(1)$	$N - 1$	Length- $O(n)$ Pauli strings	Here
Binary addressing codes	Nonlinear/nonlinear	$O(2^{n/K})$	$\log(N^K/K!)$	$(O(n))$ -controlled gates	Here
Segment codes	Linear/nonlinear	$O(n/K)$	$N / \left(1 + \frac{1}{2K}\right)$	$(O(K))$ -controlled gates	Here

At the heart of our efforts is an entirely general construction of the creation and annihilation operators in (3) given an arbitrary encoding \mathbf{e} and the corresponding decoding \mathbf{d} . As one might expect, this construction is not efficient for every choice of encoding \mathbf{e} or decoding \mathbf{d} . However, for linear encodings \mathbf{e} , but possibly nonlinear decodings \mathbf{d} , they can take on a very nice form. While in principle any classical code with the same properties can be shown to yield such mappings, we provide an appealing example of how a classical code of fixed Hamming weight [28] can be used to give an interesting mapping.

Two other approaches allow us to be more modest with the algorithmic depth in either accepting a qubit saving that is linear with N , or just saving a fixed amount of qubits for hardly any cost at all.

In previous works, trading quantum resources has been addressed for general algorithms [29], and quantum simulations [30–32]. In the two works of Moll *et al* and Bravyi *et al*, qubit requirements are reduced with a scheme that is different from ours. A qubit Hamiltonian is first obtained with e.g. the Jordan–Wigner transform, then unitary operations are applied to it in order taper qubits off successively. The paper by Moll *et al* provides a straight-forward method to calculate the Hamiltonian, that can be used to reduce the amount of qubits to a minimum, but the number of Hamiltonian terms scales exponentially with the particle number. The notion that our work is based on, was first introduced in [31] by Bravyi *et al*, for linear en- and decodings. With the generalization of this method, we hope to make the goal of qubit reduction more attainable in reducing the effort to do so. The reduction method is mediated by nonlinear codes, of which we provide different types to choose from. The transform of the Hamiltonian is straight-forward from there on, and we give explicit recipes for arbitrary codes. We can summarize our contributions as follows.

- We show that for any encoding $\mathbf{e} : \mathbb{Z}_2^{\otimes N} \rightarrow \mathbb{Z}_2^{\otimes n}$ there exists a mapping of fermionic models to quantum gates. For the special case that this encoding is linear, our procedure can be understood as a slightly modified version of the perspective taken in [24]. This gives a systematic way to employ classical codes for obtaining such mappings.
- Using particle conservation symmetry, we develop 3 types of codes that save a constant, linear and exponential amount of qubits (see table 1 and sections 3.1.1–3.1.3). An example from classical coding theory [28] is used to obtain significant qubit savings (here called the binary addressing code), at the expense of increased gate difficulty (unless the architecture would easily support multi-controlled gates).
- The codes developed are demonstrated on two examples from quantum chemistry and physics.
 1. The Hamiltonian of the well-studied hydrogen molecule in minimal basis is re-shaped into a two-qubit problem, using a simple code.
 2. A Fermi–Hubbard model on a 2×5 lattice and periodic boundary conditions in the lateral direction is considered. We parametrize and compare the sizes of the resulting Hamiltonians, as we employ different codes to save various amounts of qubits. In this way, the trade-off between qubit savings and gate complexity is illustrated (see table 2).

2. Background

To illustrate the general use of (possibly non linear) encodings to represent fermionic models, let us first briefly generalize how existing mappings can be phrased in terms of linear encodings in the spirit of [24]. Under consideration in representing the dynamics is a mapping for second-quantized Hamiltonians of the form

Table 2. Relaxing the qubit requirements for the Hamiltonian (48), where various mappings trade different amounts of qubits. The notation \oplus is used as two codes for different graphs are appended. We compare different mappings by the amount of qubits. We make comparisons by the number of Hamiltonian terms and the total weight of the resulting Pauli strings.

Mapping	Qubits	Gates	Terms
Jordan–Wigner transform	20	232	74
Bravyi–Kitaev transform	20	278	74
Checksum code \oplus Checksum code	18	260	74
Checksum code \oplus Segment code	17	4425	876
Segment code \oplus Segment code	16	9366	1838

$$\begin{aligned}
 H &= \sum_{l=0}^{\infty} \sum_{\substack{\mathbf{a} \in [N]^{\otimes l} \\ \mathbf{b} \in \mathbb{Z}_2^{\otimes l}}} h_{\mathbf{a}\mathbf{b}} \prod_{i=1}^l (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2} \\
 &= \sum_l \sum_{\substack{\mathbf{a}, \mathbf{b} \\ \text{with } h_{\mathbf{a}\mathbf{b}} \neq 0}} \hat{h}_{\mathbf{a}\mathbf{b}},
 \end{aligned} \tag{11}$$

where $h_{\mathbf{a}\mathbf{b}}$ are complex coefficients, chosen in a way as to render H Hermitian. We illustrate the use of such a mapping in the context of quantum simulation in appendix A. For our convenience, we use length- lN -ary vectors $\mathbf{a} = (a_1, \dots, a_l)^T \in [N]^{\otimes l}$ to parametrize the orbitals on which a term $\hat{h}_{\mathbf{a}\mathbf{b}}$ is acting, and write $[N] = \{1, \dots, N\}$. A similar notation will be employed for binary vectors of length l , with $\mathbf{b} = (b_1, \dots, b_l)^T \in \mathbb{Z}_2^{\otimes l}$, $\mathbb{Z}_2 = \{0, 1\}$, deciding whether an operator is a creator or annihilator by the rules $(c_i^{(\dagger)})^1 = c_i^{(\dagger)}$ and $(c_i^{(\dagger)})^0 = 1$.

Every term $\hat{h}_{\mathbf{a}\mathbf{b}}$ is a linear operation $\mathcal{F}_N \rightarrow \mathcal{F}_N$, with \mathcal{F}_N being the Fock space restricted on N orbitals, the direct sum of all possible antisymmetrized M -particle Hilbert spaces $\mathcal{H}_N^M : \mathcal{F}_N = \bigoplus_{m=0}^N \mathcal{H}_N^m$. Conventional mappings transform states of the Fock space \mathcal{F}_N into states on N qubits, carrying over all linear operations as well $\mathcal{L}(\mathcal{F}_N) \rightarrow \mathcal{L}((\mathbb{C}^2)^{\otimes N})$.

Before we start presenting conventional transformation schemes, we need to make a few remarks on transformed Hamiltonians and notations pertaining to them. First of all, we identify the set of gates $\{\mathcal{P}, \mathbb{I}\}^{\otimes n} = \{X, Y, Z, \mathbb{I}\}^{\otimes n}$ with the term Pauli strings (on n qubits). The previously mentioned Jordan–Wigner transform, obviously has the power to transform (11) into a Hamiltonian that is a weighted sum of Pauli strings on N qubits. General transforms, however, might involve other types of gates. We however have the choice to decompose these into Pauli strings. One might want to do so when using standard techniques for Hamiltonian simulation. In the following, we will denote the correspondence of second-quantized operators or states B to their qubit counterparts C by: $B \cong C$. For convenience, we will also omit identities in Pauli strings and rather introduce qubit labels, e.g. $X \otimes \mathbb{I} \otimes X = X_1 \otimes X_3 = (\bigotimes_{i \in \{1,3\}} X_i)$ and write $\mathbb{I}^{\otimes n} = \mathbb{I}$. A complete table of notations can be found in appendix G.

Consider a linear encoding of N fermionic sites into $n = N$ qubits given by a binary matrix A such that

$$|\omega\rangle = |\mathbf{e}(\nu)\rangle = |A\nu \bmod 2\rangle \cong \left(\prod_{j=1}^N (c_j^\dagger)^{\nu_j} \right) |\Theta\rangle \tag{12}$$

and A is invertible, i.e. $(AA^{-1} \bmod 2) = \mathbb{I}$. Note that in this case, the decoding given by $\nu = \mathbf{d}(\omega) = (A^{-1}\omega \bmod 2)$ is also linear. It is known that any such matrix A , subsequently also yields a mapping of the fermionic creation and annihilation operators to qubit gates [24]. To see how these are constructed, let us start by noting that they must fulfill the properties given in (3)–(6) and (2), which motivates the definition of a parity, a flip and an update set below:

1. $c_{i_m}^{(\dagger)}$ anticommutes with the first $m - 1$ operators and thus acquires phase $(-1)^{m-1}$.
2. A creation operator $c_{i_m}^\dagger$ might be absent (present) in between $c_{i_{m-1}}^\dagger$ and $c_{i_{m+1}}^\dagger$, leading the rightmost operator $c_{i_m}^{(\dagger)}$ to map the entire state to zero since $c_{i_m}|\Theta\rangle = 0$ ($c_{i_m}^\dagger c_{i_m}^\dagger = 0$).
3. Given that the state was not annihilated, the occupation of site i_m has to be changed. This means a creation operator $c_{i_m}^\dagger$ has to be added or removed between $c_{i_{m-1}}^\dagger$ and $c_{i_{m+1}}^\dagger$.

These rules tell us what the transform of an operator $c_j^{(\dagger)}$ has to inflict on a basis state (12). In order to implement the phase shift of the first rule, a series of Pauli- Z operators is applied on qubits, whose numbers are in the *parity*

set (with respect to $j \in [N]$), $P(j) \subseteq [N]$. Following the second rule we project onto the ± 1 subspace of the Z -string on qubits indexed by another $[N]$ subset, the so-called *flip set* of j , $F(j)$. The *update set* of j , $U(j) \subseteq [N]$ labels the qubits to be flipped completing the third rule using an X -string

$$(c_j^\dagger)^b (c_j)^{b+1 \bmod 2} \triangleq \frac{1}{2} \left(\bigotimes_{k \in U(j)} X_k \right) \left(\mathbb{I} - (-1)^b \bigotimes_{l \in F(j)} Z_l \right) \bigotimes_{m \in P(j)} Z_m, \quad (13)$$

with $b \in \mathbb{Z}_2$. $P(j)$, $F(j)$ and $U(j)$ depend on the matrices A and A^{-1} as well as the parity matrix R . The latter is a $(N \times N)$ binary matrix which has its lower triangle filled with ones, but not its diagonal. For the matrix entries this means $R_{ij} = \theta_{ij}$, with θ_{ij} as the discrete version of the Heaviside function

$$\theta_{ij} = \begin{cases} 0 & i \leq j \\ 1 & i > j. \end{cases} \quad (14)$$

The set members are obtained in the following fashion:

1. $P(j)$ contains all column numbers in which the j th row of matrix $(RA^{-1} \bmod 2)$ has non-zero entries.
2. $F(j)$ contains the column labels of non-zero entries in the j th row of A^{-1} .
3. $U(j)$ contains all row numbers in which the j th column of A has non-zero entries.

Note that this definition of the sets differs from their original appearance in [24, 33], where diagonal elements are not included. In this way, our sets are not disjoint, which leads to Z -cancellations and appearance of Pauli- Y operators, but we have generalized the sets for arbitrary invertible matrices, and provided a pattern for other transforms later. In fact, we recover these linear transforms from the general case in appendix F. There we also show explicitly that these operators abide by (2)–(6).

2.1. Jordan–Wigner, parity and Bravyi–Kitaev transform

As an illustration, we present popular examples of these linear transformations, note again that all of these will have $n = N$. The Jordan–Wigner transform is a special case for $A = \mathbb{I}$, leading to the direct mapping. The operator transform gives $L = O(N)$ Pauli strings as

$$(c_j^\dagger)^b (c_j)^{b+1 \bmod 2} \triangleq \frac{1}{2} (X_j + i(-1)^b Y_j) \bigotimes_{m < j} Z_m. \quad (15)$$

In the parity transform [24], we have $L = O(N)$ X -strings:

$$A^{-1} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & \vdots & \ddots & \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \quad (16)$$

$$(c_j^\dagger)^b (c_j)^{b+1 \bmod 2} \triangleq \frac{1}{2} (Z_{j-1} \otimes X_j - i(-1)^b Y_j) \bigotimes_{m=j+1}^N X_m. \quad (17)$$

The Bravyi–Kitaev transform [2] is defined by a matrix A [24, 33] that has non-zero entries according to a certain binary tree rule, achieving $L = O(\log N)$.

2.2. Saving qubits by exploiting symmetries

Our goal is to be able to trade quantum resources, which is done by reducing degrees of freedom by exploiting symmetries. For that purpose, we provide a theoretical foundation to characterize the latter.

Parity, Jordan–Wigner and Bravyi–Kitaev transforms encode all \mathcal{F}_N states and provide mappings for every $\mathcal{L}(\mathcal{F}_N)$ operator. Unfortunately, they require us to own a N -qubit quantum computer, which might be unnecessary. In fact, the only operator we want to simulate is the Hamiltonian, which usually has certain symmetries. Taking these symmetries into account enables us to perform the same task with $n \leq N$ qubits instead. Symmetries usually divide the \mathcal{F}_N into subspaces, and the idea is to encode only one of those. Let \mathcal{B} be a basis spanning a subspace $\text{span}(\mathcal{B}) \subseteq \mathcal{F}_N$ be associated with a Hamiltonian (11), where for every l , \mathbf{a} , \mathbf{b} ; $\hat{h}_{ab} : \text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B})$. Usually, Hamiltonian symmetries generate many such (distinct) subspaces. Under consideration of additional information about our problem, like particle number, parity or spin polarization, we select the correct subspace. Note that particle number conservation is by far the most prominent symmetry to

take into account. It is generated by Hamiltonians that are linear combinations of products of $c_i^\dagger c_j$ | $i, j \in [N]$. These Hamiltonians, originating from first principles, only exhibit terms conserving the total particle number; $\hat{h}_{ab} : \mathcal{H}_N^M \rightarrow \mathcal{H}_N^M$. From all the Hilbert spaces \mathcal{H}_N^M , one considers the space with the particle number matching the problem description.

These symmetries will be utilized in the next section: we develop a language that allows for encodings \mathbf{e} that reduce the length of the binary vectors $\mathbf{e}(\nu)$ as compared to ν . This means that the state ν will be encoded in $n \leq N$ qubits, since each digit saved corresponds to a qubit eliminated. As suggested by Bravyi *et al* [31], qubit savings can be achieved under the consideration of non-square, invertible matrices A . However, we will see below that using transformations based on nonlinear encodings and decodings \mathbf{d} (the inverse transform defined by A^{-1} before), we can eliminate a number of qubits that scales with the system size. For linear codes on the other hand, we find a mere constant saving.

3. General transformations

We here show how second-quantized operators and states, Hamiltonian symmetries and the fermionic basis \mathcal{B} are fused into a simple description of occupation basis states. While in this section all general ideas are presented, we would like to refer the reader to the appendices for details: to appendix B in particular, which holds the proof of the underlying techniques. Fermionic basis states are represented by binary vectors $\nu \in \mathbb{Z}_2^{\otimes N}$, with its components implicating the occupation of the corresponding orbitals. Basis states inside the quantum computer, on the other hand, are represented by binary vectors on a smaller space $\omega \in \mathbb{Z}_2^{\otimes n}$. These vectors are code words of the former ν , where the binary code connecting all ν and ω is possibly nonlinear. In the end, an instance of such a code will be sufficient to describe states and operators, in a similar way than the matrix pair (A, A^{-1}) governs the conventional transforms already presented. We now start by defining such codes and connect them to the state mappings.

Let $\text{span}(\mathcal{B})$ be a subspace of \mathcal{F}_N , as defined previously. For $n \geq \log|\mathcal{B}|$, we define two binary vector functions $\mathbf{d} : \mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2^{\otimes N}$, $\mathbf{e} : \mathbb{Z}_2^{\otimes N} \rightarrow \mathbb{Z}_2^{\otimes n}$, where we regard each component as a binary function $\mathbf{d} = (d_1, \dots, d_N)^T$ | $d_i : \mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2$. Furthermore we introduce the binary basis set $\mathcal{V} \subseteq \mathbb{Z}_2^{\otimes n}$, with

$$\nu \in \mathcal{V}, \quad \text{only if} \quad \left(\prod_{i=1}^N (c_i^\dagger)^{\nu_i} \right) |\Theta\rangle \in \mathcal{B}. \quad (18)$$

All elements in \mathcal{B} shall be represented in \mathcal{V} . If for all $\nu \in \mathcal{V}$ the binary functions \mathbf{e} and \mathbf{d} satisfy $\mathbf{d}(\mathbf{e}(\nu)) = \nu$, and for all $\omega \in \mathbb{Z}_2^{\otimes n} : \mathbf{d}(\omega) \in \mathcal{V}$, then we call the two functions encoding and decoding, respectively. An encoding-decoding pair (\mathbf{e}, \mathbf{d}) forms a code.

We thus have obtained a general form of encoding, in which qubit states only represent the subspace $\text{span}(\mathcal{B})$. The decoding, on the other hand, translates the qubit basis back to the fermionic one:

$$|\omega\rangle := \bigotimes_{j=1}^n |\omega_j\rangle \triangleq \left(\prod_{i=1}^N (c_i^\dagger)^{d_i(\omega)} \right) |\Theta\rangle. \quad (19)$$

We intentionally keep the description of these functions abstract, as the code used might be nonlinear, i.e. it cannot be described with matrices A, A^{-1} . Nonlinearity is thereby predominantly encountered in decoding rather than in encoding functions, as we will see in the examples obtained later.

For any code (\mathbf{e}, \mathbf{d}) , we will now present the transform of fermionic operators into qubit gates. Before we can do so however, two issues are to be addressed. Firstly, one observes that we cannot hope to find a transformation recipe for a singular fermionic operator $c_j^{(\dagger)}$. The reason for this is that the latter operator changes the occupation of the j th orbital. As a consequence, a state with the occupation vector ν is mapped to $(\nu + \mathbf{u}_j \bmod 2)$, where \mathbf{u}_j is the unit vector of component j ; $(u_j)_i = \delta_{ij}$. The problem is that since we have trimmed the basis, $(\nu + \mathbf{u}_j \bmod 2)$ will probably not be in \mathcal{V} , which means this state is not encoded⁴. The action of $c_j^{(\dagger)}$ is, thus, not defined. We can however obtain a recipe for the non-vanishing Hamiltonian terms \hat{h}_{ab} as they do not escape the encoded space being $(\text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B}))$ -operators. Note that this issue is never encountered in the conventional transforms, as they encode the entire Fock space.

Secondly, we are yet to introduce a tool to transform fermionic operators into quantum gates. The structure of the latter has to be similar to the linear case, as they mimic the same dynamics as presented in section 2. In general, a gate sequence will commence with some kind of projectors into the subspace with the correct occupation, as well as operators implementing parity phase shifts. The sequence should close with bit flips to update the state. The task is now to determine the form of these operators. The issue boils down to finding

⁴ ‘Unencoded state’ is actually a slightly misleading term: when we say a state $\lambda \in \mathbb{Z}_2^{\otimes n}$ is not encoded, we actually mean that it cannot be encoded and correctly decoded, so $\mathbf{d}(\mathbf{e}(\lambda)) \neq \lambda$.

operators that extract binary information from qubit states, and map it onto their phase. In other words, we need to find linear operators associated with e.g. the binary function d_j , such that it maps basis states $|\omega\rangle \rightarrow (-1)^{d_j(\omega)}|\omega\rangle$. In any case, we must recover the case of Pauli strings on their respective sets when considering linear codes. For our example, this means the linear case yields the operator $(\bigotimes_{m \in F(j)} Z_m)$. Using general codes, we are lead to define the extraction superoperation \mathfrak{X} , which maps binary functions to quantum gates on n qubits:

$$\mathfrak{X} : (\mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2) \rightarrow \mathcal{L}((\mathbb{C}^2)^{\otimes n}). \quad (20)$$

The extraction superoperator is defined for all binary vectors $\omega \in \mathbb{Z}_2^{\otimes n}$ and binary functions $f, g : \mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2$ as:

$$\begin{aligned} \mathfrak{X}[f]|\omega\rangle &= (-1)^{f(\omega)}|\omega\rangle \\ &\text{(Extraction property),} \end{aligned} \quad (21)$$

$$\begin{aligned} \mathfrak{X}[\omega \rightarrow f(\omega) + g(\omega) \bmod 2] &= \mathfrak{X}[f] \mathfrak{X}[g] \\ &\text{(Exponentiation identity),} \end{aligned} \quad (22)$$

$$\begin{aligned} \mathfrak{X}[\omega \rightarrow b] &= (-1)^b \mathbb{I} \quad | \ b \in \mathbb{Z}_2 \\ &\text{(Extracting constant functions),} \end{aligned} \quad (23)$$

$$\begin{aligned} \mathfrak{X}[\omega \rightarrow \omega_j] &= Z_j \quad | \ j \in [n] \\ &\text{(Extracting linear functions),} \end{aligned} \quad (24)$$

$$\begin{aligned} \mathfrak{X}\left[\omega \rightarrow \prod_{j \in \mathcal{S}} \omega_j\right] &= C^k \text{PHASE}(i_1, \dots, i_{k+1}) \\ &\text{with } \mathcal{S} = \{i_s\}_{s=1}^{k+1} \subseteq [n], \quad k \in [n-1] \\ &\text{(Extracting non-linear functions).} \end{aligned} \quad (25)$$

Note that the first two properties imply that the operators $\mathfrak{X}[f]$, $\mathfrak{X}[g]$ commute and all operators are diagonal in the computational basis. Given that binary functions have a polynomial form, we are now able to construct operators by extracting every binary function possible, for example

$$\begin{aligned} \mathfrak{X}[\omega \rightarrow 1 + \omega_1 + \omega_1\omega_2 \bmod 2] \\ = \mathfrak{X}[\omega \rightarrow 1] \mathfrak{X}[\omega \rightarrow \omega_1] \mathfrak{X}[\omega \rightarrow \omega_1\omega_2] \end{aligned} \quad (26)$$

$$= -Z_1 \text{CPHASE}(1, 2). \quad (27)$$

We firstly we have used (22) to arrive at (26), and then reach (27) by applying the properties (23)–(25) to the respective sub-terms. This might however not be the final Hamiltonian, since the simulation algorithm might require us to reformulate the Hamiltonian as a sum of weighted Pauli strings [4, 5]. In that case, need to decompose all controlled gates. The cost for this decomposition is an increase in the number of Hamiltonian terms, for instance we find $\text{CPHASE}(i, j) = \frac{1}{2}(\mathbb{I} + Z_i + Z_j - Z_i \otimes Z_j)$. In general, (24) and (25) can be replaced by an adjusted definition:

$$\begin{aligned} \mathfrak{X}\left[\omega \rightarrow \prod_{j \in \mathcal{S}} \omega_j\right] &= \mathbb{I} - 2 \prod_{j \in \mathcal{S}} \frac{1}{2}(\mathbb{I} - Z_j) \quad \left| \ \mathcal{S} \subseteq [n] \right. \\ &\text{(Extracting non-constant functions).} \end{aligned} \quad (28)$$

We will be able to define the operator mappings introducing the parity and update functions, \mathbf{p} and ε^q :

$$\mathbf{p} : \mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2^{\otimes N}, \quad p_j(\omega) = \sum_{i=1}^{j-1} d_i(\omega) \bmod 2, \quad (29)$$

$$\begin{aligned} \varepsilon^q : \mathbb{Z}_2^{\otimes n} &\rightarrow \mathbb{Z}_2^{\otimes N}, \quad \text{with } \mathbf{q} \in \mathbb{Z}_2^{\otimes N} \\ \varepsilon^q(\omega) &= \mathbf{e}(\mathbf{d}(\omega) + \mathbf{q} \bmod 2) + \omega \bmod 2. \end{aligned} \quad (30)$$

Finally, we have collected all the means to obtain the operator mapping for weight- l operator sequences as they occur in (11):

$$\begin{aligned} \prod_{i=1}^l (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2} &\triangleq \mathcal{U}^a \left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{a_v a_w}} \right) \\ &\times \prod_{x=1}^l \frac{1}{2} \left(\mathbb{I} - \left[\prod_{y=x+1}^l (-1)^{\delta_{a_x a_y}} \right] (-1)^{b_x} \mathfrak{X}[d_{a_x}] \right) \mathfrak{X}[p_{a_x}], \end{aligned} \quad (31)$$

where θ_{ij} is defined in (14) and δ_{ij} is the Kronecker delta. In this expression, we find various projectors, parity operators with corrections for occupations that have changed before the update operator is applied. The update operator \mathcal{U}^a , is characterized by the $\mathbb{Z}_2^{\otimes N}$ -vector $\mathbf{q} = \sum_{i=1}^I \mathbf{u}_{a_i} \bmod 2$

$$\mathcal{U}^a = \sum_{\mathbf{t} \in \mathbb{Z}_2^{\otimes n}} \left[\bigotimes_{i=1}^n (X_i)^{t_i} \right] \prod_{j=1}^n \frac{1}{2} (\mathbb{I} + (-1)^{t_j} \mathfrak{X}[\varepsilon_j^a]). \quad (32)$$

This is a problem: when summing over the entire $\mathbb{Z}_2^{\otimes n}$, one has to expect an exponential number of terms. As a remedy, one can arrange the resulting operations into controlled gates, or rely on codes with a linear encoding. If the encoding can be defined using a binary $(n \times N)$ -matrix A , $\mathbf{e}(\boldsymbol{\nu}) = (A\boldsymbol{\nu} \bmod 2)$, the update operator reduces to

$$\mathcal{U}^a = \bigotimes_{i=1}^n (X_i)^{\sum_j A_{ij} q_j \bmod 2}. \quad (33)$$

In appendix B, we show that (31)–(33) satisfy the conditions (2)–(6). Note that the update operator is also important for state preparation: let us assume that our qubits are initialized all in their zero state, $(\bigotimes_{i \in [n]} |0\rangle)$, then the fermionic basis state associated with the vector $\boldsymbol{\nu}$ is obtained by applying the update operator \mathcal{U}^a . Here the vector \mathbf{a} contains all occupied orbitals, such that $\mathbf{q} = \boldsymbol{\nu}$. Even for nonlinear encodings the state preparation can be done with Pauli strings: as the initial state is a product state of all zeros, we can replace operators $\mathfrak{X}[\boldsymbol{\omega} \rightarrow \prod_{i \in \mathcal{S} \subseteq [n]} \omega_i]$ by \mathbb{I} .

In the following we will turn our attention to the most fruitful symmetry to take into account: particle conservation symmetry. While code families accounting for this symmetry are explored in the next subsection, alternatives to the mapping of entire Hamiltonian terms are discussed for such codes in appendix C.

3.1. Particle-number conserving codes

In the following, we will present three types of codes that save qubits by exploiting particle number conservation symmetry, and possibly the conservation of the total spin polarization. Particle-number conserving Hamiltonians are highly relevant for quantum chemistry and problems posed from first principles. We therefore set out to find codes in which $\boldsymbol{\nu} \in \mathcal{V}$ have a constant Hamming weight $w_H(\boldsymbol{\nu}) = K$. Since the Hamming weight is defined as $w_H(\boldsymbol{\nu}) = \sum_m \nu_m$, it yields the total occupation number for the vectors $\boldsymbol{\nu}$. In order to simulate systems with a fixed particle number, we are thus interested to find codes that implement code words of constant Hamming weight. Note that the fixed Hamming weight K does not necessarily need to coincide with the total particle number M . A code with the latter property might also be interesting for systems with additional symmetries. Most importantly, we have not taken into account the spin-multiplicity yet. As the particles in our system are fermions, every spatial site will typically have an even number of spin configurations associated with it. Orbitals with the same spin configurations naturally denote subsets of the total amount of orbitals, much like the suits in a card deck. An absence of magnetic terms as well as spin-orbit interactions leaves the Hamiltonian to conserve the number of particles inside all those suits. Consequently, we can append several constant-weight codes to each other. Each of those subcodes encodes thereby the orbitals inside one suit. In electronic system with only Coulomb interactions for instance, we can use two subcodes $(\mathbf{e}^\diamond, \mathbf{d}^\diamond)$ and $(\mathbf{e}^\spadesuit, \mathbf{d}^\spadesuit)$, to encode all spin-up, and spin-down orbitals, respectively. The global code (\mathbf{e}, \mathbf{d}) , encoding the entire system, is obtained by appending the subcode functions e.g. $\mathbf{d}(\boldsymbol{\omega}^1 \oplus \boldsymbol{\omega}^2) = \mathbf{d}^\diamond(\boldsymbol{\omega}^1) \oplus \mathbf{d}^\spadesuit(\boldsymbol{\omega}^2)$. Appending codes like this will help us to achieve higher savings at a lower gate cost.

The codes that we now introduce (see also again table 1), fulfill the task of encoding only constant-weight words differently well. The larger \mathcal{V} , the less qubits will be eliminated, but we expect the resulting gate sequences to be more simple. Although not just words of that weight are encoded, we treat K as a parameter—the targeted weight.

3.1.1. Checksum codes

A slim, constant amount of qubits can be saved with the following $n = N - 1$, affine linear codes. Checksum codes encode all the words with either even or odd Hamming weight. As this corresponds to exactly half of the Fock space, one qubit is eliminated. This means we disregard the last component when we encode $\boldsymbol{\nu}$ into words with one digit less. The decoding function then adds the missing component depending on the parity of the code words. The code for K odd is defined as

$$\mathbf{d}(\boldsymbol{\omega}) = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 1 & \cdots & 1 \end{bmatrix} \boldsymbol{\omega} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \bmod 2, \quad (34)$$

$$\mathbf{e}(\boldsymbol{\nu}) = \begin{bmatrix} 1 & & 0 \\ & \ddots & \vdots \\ & & 1 & 0 \end{bmatrix} \boldsymbol{\nu} \bmod 2. \quad (35)$$

In the even- K version, the affine vector \mathbf{u}_N , added in the decoding, is removed. Since encoding and decoding function are both at most affine linear, the extracted operators will all be Pauli strings, with at most a minus sign. The advantage of the checksum codes is that they do not depend on K . They can be used even in cases of smaller saving opportunities, like $K \approx N/2$. We can employ these codes even for Hamiltonians that conserve only the Fermion parity. This makes them important for effective descriptions of superconductors [34].

3.1.2. Codes with binary addressing

We present a concept for heavily nonlinear codes for large qubit savings, $n = \lceil \log(N^K/K!) \rceil$, [28]. In order to conserve the maximum amount of qubits possible, we choose to encode particle coordinates as binary numbers in $\boldsymbol{\omega}$. To keep it simple, we here consider the example of weight-one binary addressing codes, and refer the reader to appendix D for $K > 1$. In $K = 1$, we recognize the qubit savings to be exponential, so consider $N = 2^n$. Encoding and decoding functions are defined by means of the binary enumerator, $\text{bin} : \mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}$, with $\text{bin}(\boldsymbol{\omega}) = \sum_{j=1}^n 2^{j-1} \omega_j$

$$d_j(\boldsymbol{\omega}) = \prod_{i=1}^n (\omega_i + 1 + q_i^j) \bmod 2, \quad (36)$$

$$\mathbf{e}(\boldsymbol{\nu}) = [q^1 | q^2 | \dots | q^{2^n}] \boldsymbol{\nu} \bmod 2, \quad (37)$$

where $q^j \in \mathbb{Z}_2^{\otimes n}$ is implicitly defined by $\text{bin}(q^j) + 1 = j$. An input $\boldsymbol{\omega}$ will by construction render only the j th component of (36) non-zero, when $q^j = \boldsymbol{\omega}$ ⁵.

The exponential qubit saving comes at a high cost: the product over each component of $\boldsymbol{\omega}$ implies multi-controlled gates on the entire register. This is likely to cause connectivity problems. Note that decomposing the controlled gates will in general be practically prohibited by the sheer amount of resulting terms. On top of those drawbacks, we also expect the encoding function to be nonlinear for $K > 1$.

3.1.3. Segment codes

We introduce a type of scaleable $n = \lceil N / (1 + \frac{1}{2K}) \rceil$ codes to eliminate a linear amount of qubits. The idea of segment codes is to cut the vectors $\boldsymbol{\nu}$ into smaller, constant-size vectors $\hat{\boldsymbol{\nu}}^i \in \mathbb{Z}_2^{\otimes \hat{N}}$, such that $\boldsymbol{\nu} = \bigoplus_i \hat{\boldsymbol{\nu}}^i$. Each such segment $\hat{\boldsymbol{\nu}}^i$ is encoded by a subcode. Although we have introduced the concept already, this segmentation is independent from our treatment of spin ‘suits’. In order to construct a weight K global code, we append several instances of the same subcode. Each of these subcodes is defined on \hat{n} qubits, encoding $\hat{N} = \hat{n} + 1$ orbitals. We deliberately have chosen to only save one qubit per segment in order to keep the segment size $\hat{N}(K)$ small.

We now turn our attention to the construction of these segment codes. As shown in appendix E, the segment sizes can be set to $\hat{n} = 2K$ and $\hat{N} = 2K + 1$. As the global code is supposed to encode all $\boldsymbol{\nu} \in \mathbb{Z}_2^{\otimes N}$ with Hamming weight K , each segment must encode all vectors from Hamming weight zero up to weight K . In this way, we guarantee that the encoded space contains the relevant, weight K subspace. This construction follows from the idea that each block contains equal or less than K particles, but might as well be empty. For each segment, the following de- and encoding functions are found for $\hat{\boldsymbol{\omega}} \in \mathbb{Z}_2^{\otimes \hat{n}}$, $\hat{\boldsymbol{\nu}} \in \mathbb{Z}_2^{\otimes \hat{N}}$:

$$\hat{\mathbf{d}}(\hat{\boldsymbol{\omega}}) = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 0 & \dots & 0 \end{bmatrix} \hat{\boldsymbol{\omega}} + f(\hat{\boldsymbol{\omega}}) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \bmod 2, \quad (38)$$

$$\hat{\mathbf{e}}(\hat{\boldsymbol{\nu}}) = \begin{bmatrix} 1 & & 1 \\ & \ddots & \vdots \\ & & 1 & 1 \end{bmatrix} \hat{\boldsymbol{\nu}} \bmod 2, \quad (39)$$

where $f : \mathbb{Z}_2^{\otimes \hat{n}} \rightarrow \mathbb{Z}_2$ is a binary switch. The switch is the source of nonlinearity in these codes. On an input $\hat{\boldsymbol{\omega}}$ with $w_H(\hat{\boldsymbol{\omega}}) > K$, it yields one, and zero otherwise.

There is just one problem: segment codes are not suitable for particle-number conserving Hamiltonians, according to the definition of the basis \mathcal{B} , that we would have for segment codes. The reason for this is that we have not encoded all states with $w_H(\boldsymbol{\nu}) > K$. In this way, Hamiltonian terms \hat{h}_{ab} that exchange occupation numbers between two segments, can map into unencoded space. We can, however, adjust these terms, such that

⁵ For better or worse we have used the binary representation of the orbital indexes. We could however employ any other counting method, i.e. any injective mapping that relates a binary vector representing a qubit basis state to an integer labeling an orbital.

they only act non-destructively on states with at most K particles between the involved segment. This does not change the model, but aligns the Hamiltonian with the necessary condition that we have on \mathcal{B} , $\hat{h}_{ab} : \text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B})$. This is discussed in detail appendix E, where we also provide an explicit description of the binary switch mentioned earlier.

Using segment codes, the operator transforms will have multi-controlled gates as well: the binary switch is nonlinear. However, gates are controlled on at most an entire segment, which means there is no gate that acts on more than $2K$ qubits. This an improvement in gate locality, as compared to binary addressing codes.

4. Examples

4.1. Hydrogen molecule

In this subsection, we will demonstrate the Hamiltonian transformation on a simple problem. Choosing a standard example, we draw comparison with other methods for qubit reduction. As one of the simplest problems, the minimal electronic structure of the hydrogen molecule has been studied extensively for quantum simulation [3, 4] already. We describe the system as two electrons on 2 spatial sites. Because of the spin-multiplicity, we require 4 qubits to simulate the Hamiltonian in conventional ways. Using the particle conservation symmetry of the Hamiltonian, this number can be reduced. The Hamiltonian also lacks terms that mix spin-up and -down states, with the total spin polarization known to be zero. Taking into account these symmetries, one finds a total of 4 fermionic basis states: $\mathcal{V} = \{(0, 1, 0, 1), (0, 1, 1, 0), (1, 0, 0, 1), (1, 0, 1, 0)\}$. These can be encoded into two qubits by appending two instances of a $(N = 2, n = 1, K = 1)$ -code. The global code is defined as :

$$\mathbf{d}(\omega) = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \omega + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \text{ mod } 2, \quad (40)$$

$$\mathbf{e}(\nu) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \nu \text{ mod } 2. \quad (41)$$

The physical Hamiltonian

$$\begin{aligned} H = & -h_{11}(c_1^\dagger c_1 + c_3^\dagger c_3) - h_{22}(c_2^\dagger c_2 + c_4^\dagger c_4) \\ & + h_{1331} c_1^\dagger c_3^\dagger c_3 c_1 + h_{2442} c_2^\dagger c_4^\dagger c_4 c_2 \\ & + h_{1221}(c_1^\dagger c_4^\dagger c_4 c_1 + c_3^\dagger c_2^\dagger c_2 c_3) \\ & + (h_{1221} - h_{1212})(c_1^\dagger c_2^\dagger c_2 c_1 + c_3^\dagger c_4^\dagger c_4 c_3) \\ & + h_{1212}(c_1^\dagger c_4^\dagger c_3 c_2 + c_2^\dagger c_3^\dagger c_4 c_1) \\ & + h_{1212}(c_1^\dagger c_3^\dagger c_4 c_2 + c_2^\dagger c_4^\dagger c_3 c_1), \end{aligned} \quad (42)$$

is transformed into the qubit Hamiltonian

$$g_1 \mathbb{I} + g_2 X_1 \otimes X_2 + g_3 Z_1 + g_4 Z_2 + g_5 Z_1 \otimes Z_2. \quad (43)$$

The real coefficients g_i are formed by the coefficients h_{ijkl} of (42). After performing the transformation, we find

$$g_1 = -h_{11} - h_{22} + \frac{1}{2}h_{1221} + \frac{1}{4}h_{1331} + \frac{1}{4}h_{2442}, \quad (44)$$

$$g_2 = h_{1212}, \quad (45)$$

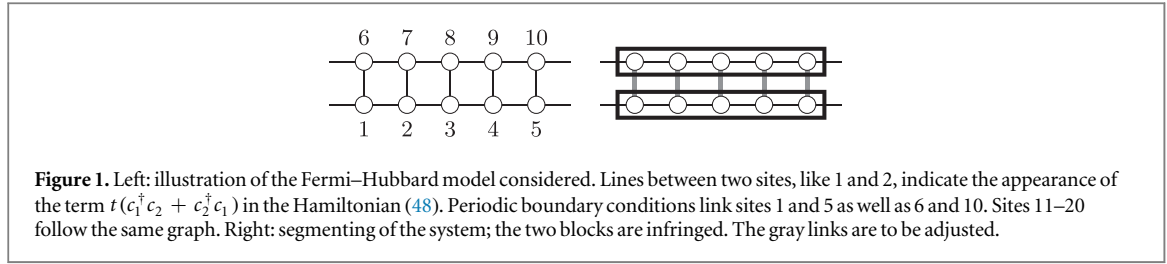
$$g_3 = g_4 = \frac{1}{2}h_{11} - \frac{1}{2}h_{22} + -\frac{1}{4}h_{1331} + \frac{1}{4}h_{2442}, \quad (46)$$

$$g_5 = -\frac{1}{2}h_{1221} + \frac{1}{4}h_{1331} + \frac{1}{4}h_{2442}. \quad (47)$$

In previous works, conventional transforms have been applied to that problem Hamiltonian. Afterwards, the resulting 4-qubit-Hamiltonian has been reduced by hand in some way. In [11], the actions on two qubits are replaced with their expectation values after inspection of the Hamiltonian. In [30], on the other hand, the Hamiltonian is reduced to two qubits in a systematic fashion. Finally, the case is revisited in [31], where the problem is reduced below the combinatorial limit to one qubit. The latter two attempts have used Jordan–Wigner, the former the Bravyi–Kitaev transform first.

4.2. Fermi–Hubbard model

We present another example to illustrate the trade-off between qubit number and gate cost as well as circuit depth. For that purpose, we consider a simple toy Hamiltonian and demonstrate that a reduction of qubit requirements is theoretically possible. Although we do not want to claim that this scenario is realistic, we present



a simple cost model with it, that hints the potential up-scaling of circuit depth and simulation cost, as the number of qubits decreases: we therefore consider the total sum of Pauli lengths of every term, which gives us an idea of the number of two-qubit gates required, and the number of Hamiltonian terms, as we decompose controlled gates (28), which should give us an idea of possible T-gate requirements and simulation depth. Let us start now to describe the model. We consider a small lattice with periodic boundary conditions in the lateral direction. The system shall contain 10 spatial sites, doubled by the spin-multiplicity. The problem Hamiltonian is

$$H = -t \sum_{\langle i,j \rangle \in E} (c_i^\dagger c_j + c_j^\dagger c_i) + U \sum_{j=1}^{10} c_j^\dagger c_j c_{10+j}^\dagger c_{10+j}, \quad (48)$$

with its real coefficients t, U . It exhibits hopping terms along the edges E of the graph in figure 1. The sketch on the left of this figure shows the connection graph of the first 10 orbitals. The other 10 orbitals are connected in the same fashion, and each such site is interacting with its counterpart from the other graph. We aim to populate this model with four Fermions, where the total spin polarization is zero. Two conventional transforms and two transforms based on our codes are compared by the amount of qubits necessary, as well as the size of the transformed Hamiltonian. Note that besides eigenenergies, one might also be interested in obtaining the values of correlation functions, e.g. $\langle c_i^\dagger c_j \rangle$, which is done by measuring (qubit) operators obtained with the transform (48). The only difference is that if a correlator maps into unencoded space, it is to be set to zero. As benchmarks, we decompose controlled gates and count the number of resulting Pauli strings. The sum of their total weight constitutes the gate count. Having these two disconnected graphs is an invitation to us to append two codes acting on sites 1–10 and 11–20 respectively. for this example, we consider the following codes:

1. Jordan–Wigner and Bravyi–Kitaev transform: for comparison, we employ these conventional transforms on our system, with which we do not save qubits. The resulting terms are best obtained by the transforming every Fermion operator in (48) by (13), where the flip, parity and update sets, $F(j), P(j), U(j)$ are determined by the choice of matrices A and A^{-1} , which are binary tree matrices in the the case of the Bravyi–Kitaev transform, and identity matrices for the Jordan–Wigner transform.
2. Checksum code \oplus checksum code: knowing that the particle number is conserved, and that spin cannot be flipped, we are free to save 2 qubits in constraining the parity of both, spin-up and -down particles, alike. This is done in appending two ($N = 10$) checksum codes, where each that acts on only spin-up (spin-down) orbitals, so indices 1–10 (11–20). The code resulting from appending two even checksum codes is linear, and encoding and decoding function feature the matrices

$$\left[\begin{array}{ccc|ccc} 1 & & & 0 & & \\ & \ddots & & \vdots & & \\ & & 1 & 0 & & \\ \hline & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 & 0 \end{array} \right], \left[\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline 1 & \dots & 1 & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 & 1 & \dots & 1 \end{array} \right]. \quad (49)$$

However, as not the entire Fock space is encoded, we need to perform the operator transform according to (31), where the update operator is defined by (33), where A refers here to the first matrix.

3. Segment code \oplus segment code: knowing the particle number in one ‘spin suite’ to be 2, we can for both, spin-up and -down orbitals, append two $K = 2$ segment codes to each other. This equals a total of 4 segment codes, saving 4 qubits. The resulting global code (\mathbf{e}, \mathbf{d}) is defined by

$$e\left(\bigoplus_{i=1}^4 \hat{\nu}^i\right) = \bigoplus_{i=1}^4 \hat{e}(\hat{\nu}^i), \quad (50)$$

$$d(\omega) = \left(\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & & & 1 \\ & \ddots & & \vdots \\ & & 1 & 1 \end{bmatrix} \right) \omega \bmod 2, \quad (51)$$

where \hat{e} are the encodings of the subcodes (39), and $\hat{\nu}^i$ are occupations on the segments of the total orbital vector $\nu = \bigoplus_i \hat{\nu}^i$. These segments are formed as suggested by the right-hand side of figure 1. For details on the decoding functions and Hamiltonian adjustments, please consider appendix E. The Hamiltonian transform is in the end carried out again by (31) and (33).

4. Checksum code \oplus segment code: a compromise between the above, in which the spin-up orbitals are transformed via a checksum code, and the spin-down orbitals are transformed via two segment codes. The global code used for the Hamiltonian transformation is the appendage of an (even-weight, $N = 10$) checksum code and two ($K = 2$) segment codes, including Hamiltonian adjustments on the spin-down orbitals.

Note that from the combinatorial perspective, we could encode the problem with 11 qubits. However, if we append two $K = 2$ binary addressing codes to each other, the resulting Hamiltonian is on 14 qubits already. The problem is that the resulting Hamiltonian for this case cannot be expressed with decomposed controlled gates due to the high number of resulting terms.

Indeed, table 2 suggests that decomposing the controlling gates might easily lead to very large Hamiltonians with a multitude of very small terms. The gate decomposition appears therefore undesirable. We in general recommend to rather decompose large controlled gates as shown in [35]. However, one also notices that an elimination of up to two qubits comes at a low cost: the amount of gates is not higher than in the Bravyi–Kitaev transform. As soon as we employ segment codes on the other hand, the Hamiltonian complexity rises with the amount of qubits eliminated.

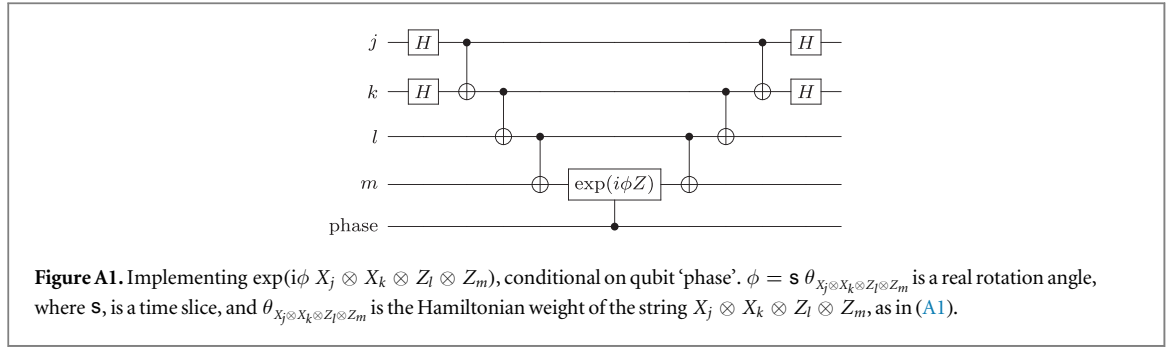
5. Conclusion and future work

In this work, we have introduced new methods to reduce the number of qubits required for simulating fermionic systems in second quantization. We see the virtue of the introduced concepts in the fact that it takes into account symmetries on a simple but non-abstract level. We merely concern ourselves with objects as simple as binary vectors, but attribute the physical interpretation of orbital occupations to them. At this level, the mentioned symmetries are easy to apply and exploit. The accounting for the complicated antisymmetrization of the many-body wave function on the other hand is done in the fermionic operators, which to transform we have provided recipes for. In these operator transforms we see room for improvement: we for instance lack a proper gate composition for update operators of nonlinear encodings at this point. We on the other hand have the extraction superoperator \mathfrak{X} return only conventional (multi)-controlled phase gates. Nonlinear codes would on the other hand benefit from a gate set that includes gates with negative control, i.e. with the (-1) eigenvalue conditioned on $|0\rangle$ eigenspaces of certain qubits involved. We consider our work to be relevant for quantum simulation with near-term devices, with a limited number of qubits at disposal. Remarks about asymptotic scaling are thus missing in this work, but would be interesting. Also, we have centered our investigations around quantum computers with qubits. The idea behind the generalized operator transforms, however, can possibly be adapted to multi-level systems (qudits). The operator transforms of segment and binary addressing codes, for instance, might simplify in such a setup, if generalized Pauli operators are available in some form.

Apart from the codes presented, we have laid the foundation for the reader to invent their own. As supplementary material, we include a program to transform arbitrary Hamiltonians from a second-quantized form into Pauli string form, using user-defined codes. In this way we hope that in the long term, many more entries will be added to table 1. The extension of this work to a more general setting for symmetries, in which the latter are generated by groups or sets of operators that commute with the Hamiltonian, is an open task. Furthermore, we are certain that table 1 can be extended into another way: gate relaxations for transforms with $n > N$ have already been shown [2, 23, 26, 36], and we are currently working in that direction.

Acknowledgments

We would like to thank Kenneth Goodenough, Victoria Lipinska, Thinh Le Phuc and Valentina Caprara Vivoli for helpful discussions on how to present our results. We would also like to thank CWJ Beenakker for his



support. MS was supported by the Netherlands Organization for Scientific Research (NWO/OCW) and an ERC Synergy Grant. SW was supported by STW Netherlands, an NWO VIDI Grant and an ERC Synergy Grant.

Appendix A. On quantum simulation

At this point, we discuss quantum simulation in the context of our transformations. Amongst other things, we describe the most simple algorithm for Hamiltonian simulation, and proceed by investigating feasibility issues with our transforms. Let us start by explaining how this work fits into the larger frame.

The transformations we have developed are going to be useful to trade quantum resources for quantum simulation of fermionic systems, independent from the concrete quantum algorithms chosen for simulation of the problem. For those problems from quantum chemistry and many-body physics we are usually given a fermionic system and its Hamiltonian. One is then to determine the system's ground state and ground state energy, sometimes parts of its spectrum. Where classical computation is infeasible, we simulate the system inside a quantum computer, on which the problem can be solved with existing algorithms. With either transform (see table 1), the fermionic system is therefore mapped to a system of n qubits. With the operator transform, H turns into \mathbf{H} , a sum of weighted $\mathcal{L}((\mathbb{C}^2)^{\otimes n})$ gates, Pauli strings at best. We then apply algorithms like quantum phase estimation [3, 37–39], variational quantum eigensolvers [9, 11, 12, 40], or adiabatic simulations [6]. All of those algorithms receive ansatz states as inputs and in some way prepare (eigen-) states, while also outputting their energy. The ground state is the state with the lowest energy, and can be then manipulated as it is inside the quantum registers after the simulation. For the remainder of this appendix, we discuss implications of the simulation algorithms onto our transforms. Thus we outline some principles, these algorithms rely on: algorithms might require us to simulate the time evolution of our encoded system according to \mathbf{H} . For that purpose, we need to know how to transform the time evolution operator $\exp(i\mathbf{H}t)$, where t is a time step, into gate sequences. Maybe we even need to apply those evolution conditionally, means as an operation controlled on an auxiliary qubit (register). We thus need to embed \mathbf{H} into an algorithm for Hamiltonian simulation.

Let us now be a bit more concrete, and select such an algorithm. Despite the wide range of theoretical proposals for Hamiltonian simulation algorithms [41–43], only the perhaps simplest scheme appears to be experimentally feasible for digital quantum simulation at the moment. Note that it can only be applied to Hamiltonians that are a sum of Pauli strings with real weights

$$\mathbf{H} = \sum_{\sigma \in \{X, Y, Z, \mathbb{I}\}^{\otimes n}} \theta_{\sigma} \times \sigma \quad \text{with all } \theta_{\sigma} \in \mathbb{R}. \quad (\text{A1})$$

The idea is to approximate $\exp(i\mathbf{H}t)$, by sequences of the exponentiated Pauli strings $\exp(i\theta_{\sigma} \mathbf{s} \sigma)$, where \mathbf{s} is a time slice of t . This method is commonly referred to as Trotterization. The numbers, signs and values of the time slices \mathbf{s} , as well as the ordering of the exponentiated strings, govern the error of the simulation—strategies to minimize that error can be learned from the works of Suzuki [44, 45]. Note that we do not specify whether the Hamiltonian simulation is performed in an analog or digital fashion, however, not all strings σ are feasible to be implemented in an analog fashion. The digital gadget for the exponentiation of Pauli strings, on the other hand, is well known [46]. See figure A1 for an example. We are therefore able to approximately perform a (conditional) simulated time evolution with \mathbf{H} of the form (A1). Using algorithms like variational eigensolvers, where we do not simulate the time evolution but estimate the Hamiltonian expectation value by measuring its terms, we are in principle not tied to the structure of (A1). However, it is more convenient. Equation (A1) gives us two constraints on how to transform (11).

The first constraint is that we need to decompose every fermionic operator into Pauli strings, using (28). The total number of Pauli strings resulting can be a problematically high when the underlying codes are highly nonlinear. For Trotterization that means a tremendous increase in length due to the abundance of sequenced Pauli string gadgets, many of them with very small rotation angles (ϕ in figure A1).

The second constraint seems trivial at first: in order to simulate a Hamiltonian, it has to be hermitian. More precisely, it has to be hermitian on the entire $(\mathbb{C}^2)^{\otimes n}$, so the coefficient θ_σ have to be real. We on the other hand might not even need the entire $(\mathbb{C}^2)^{\otimes n}$ to encode our physical system. Non-hermiticities, meaning complex coefficients θ_σ , can occur whenever one is careless with the remainder of the qubit space, when the code space is left or states are encoded in an ambiguous way. We here list a few pitfalls that can cause non-hermitian terms to occur after the transform and discuss how to avoid them.

- Issues may be caused by codes that are not one-to-one. A one-to-one code (\mathbf{e}, \mathbf{d}) has the property: $\mathbf{e}(\mathbf{d}(\omega)) = \omega$ for all $\omega \in \mathbb{Z}_2^{\otimes n}$. Although we have excluded the one-to-one property from the definition of the codes (taking into account the next item), it assures the hermiticity of the transformed Hamiltonian.
- The encoded basis \mathcal{B} has a size that is in between 2^n and 2^{n-1} , so n qubits provide too much Hilbert space by default. However, we can always add a state to the basis that is mapped to zero by all terms \hat{h}_{ab} . This state, represented by ν , can have several partners on the code space ω , for which $\mathbf{d}(\omega) = \nu$ (i.e. not be mapped one-to-one). For example for particle-number conserving Hamiltonians, we can balance these dimensional mismatches using the vacuum state in such a way, since $c_i^\dagger c_j |\Theta\rangle = 0$.
- We encounter this problem when using a code with a Hamiltonian, that is not feasible with it. The segment codes for instance are feasible only for certain adjusted particle-number-conserving Hamiltonians, as we shall see in appendix E.

Appendix B. General operator mappings

The goal of this appendix is to verify that the fermionic mode is accurately represented by our qubit system. This is divided into three steps: step one is to analyze the action of Hamiltonian terms on the fermionic basis. In the second step, we verify parity and projector parts of (31) to work like the original operators in step one, disregarding the occupational update for a moment. Conditions for this state update are subsequently derived. The update operator (32) is shown to fulfill these conditions in the third step, thus concluding the proof.

B.1. Hamiltonian dynamics

In order to verify that the gate sequences (31) are mimicking the Hamiltonian dynamics adequately, we verify that the resulting terms have the same effect on the Hamiltonian basis. This is done on the level of second quantization with respect to the notation (18): no transition into a qubit system is made. This step serves the sole purpose to quantify the effect of the Hamiltonian terms on the states. To that end, we begin by studying the effect of a singular fermionic operator $c_j^{(\dagger)}$ on a pure state, before considering an entire term \hat{h}_{ab} on a state in \mathcal{B} . As a preliminary, we note that (3)–(6) follow directly from (2), when considering that

$$c_j c_j = c_j^\dagger c_j^\dagger = c_j |\Theta\rangle = 0. \quad (\text{B1})$$

The relations (3)–(6) indicate how singular operators act on pure states in general. We now become more specific and apply these rules to a state $(\prod_i (c_i^\dagger)^{\nu_i}) |\Theta\rangle$, that is not necessarily in \mathcal{B} , but is described by an occupation vector $\nu \in \mathbb{Z}_2^{\otimes N}$. The effect of an annihilation operator on such a state is considered first:

$$c_j \left[\prod_{i=1}^N (c_i^\dagger)^{\nu_i} \right] |\Theta\rangle = \left[\prod_{i<j} (-c_i^\dagger)^{\nu_i} \right] c_j (c_j^\dagger)^{\nu_j} \left[\prod_{k>j} (c_k^\dagger)^{\nu_k} \right] |\Theta\rangle \quad (\text{B2})$$

$$= \left[\prod_{i<j} (-c_i^\dagger)^{\nu_i} \right] \frac{1}{2} [1 - (-1)^{\nu_j}] \left[\prod_{k>j} (c_k^\dagger)^{\nu_k} \right] |\Theta\rangle \quad (\text{B3})$$

$$= \left[\prod_{i<j} (-1)^{\nu_i} \right] \frac{1}{2} [1 - (-1)^{\nu_j}] \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k + \delta_{jk} \bmod 2} \right] |\Theta\rangle. \quad (\text{B4})$$

A short explanation on what has happened: in (B2), c_j has anticommuted with all creation operator c_i^\dagger that have indexes $i < j$. Depending on the component ν_j , a creation operator c_j^\dagger might now be to the right of the annihilator c_j . If the creation operator is not encountered, we may continue the anticommutations of c_j until it meets the vacuum and annihilates the state by $c_j |\Theta\rangle = 0$. Using the anticommutation relations (2), we therefore replace $c_j (c_j^\dagger)^{\nu_j}$ with $\frac{1}{2} [1 - (-1)^{\nu_j}]$ when going from (B2) to (B3). Finally, the terms are rearranged in (B4): conditional sign changes of the anticommutations are factored out of the new state with an occupation that is now described by the binary vector $(\nu + \mathbf{u}_j \bmod 2)$ rather than ν . When considering to apply a creation operator c_j^\dagger on the former state, the result is similar. Alone at step (B3), we have to replace $c_j^\dagger (c_j^\dagger)^{\nu_j}$ by

$\frac{1}{2}[1 + (-1)^{\nu_j}]$ instead, as now the case of appearance of the creation operator leads to annihilation: $c_j^\dagger c_j^\dagger = 0$. We thus find

$$c_j^\dagger \left[\prod_{i=1}^N (c_i^\dagger)^{\nu_i} \right] |\Theta\rangle = \left[\prod_{i<j} (-1)^{\nu_i} \right] \frac{1}{2} [1 + (-1)^{\nu_j}] \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k + \delta_{jk} \bmod 2} \right] |\Theta\rangle. \quad (\text{B5})$$

We now turn our attention to the actual goal, effect of a Hamiltonian term from (11) on a state in \mathcal{B} (this means its occupation vector ν is in \mathcal{V}). We therefore consider a generic operator sequence $\prod_{i=1}^l (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2}$, parametrized by some N -ary vector $\mathbf{a} \in [N]^{\otimes l}$ and a binary vector $\mathbf{b} \in \mathbb{Z}_2^{\otimes l}$, for some length l . With (B4) and (B5), we now have the means to consider the effect such a sequence of annihilation and creation operators. The two relations will be repeatedly utilized in an inductive procedure, as every single operator $(c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2}$ of $\prod_{i=1}^l (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2}$ will act on a basis state, one after another. The state's occupation is updated after every such operation. For convenience, we define:

$$\nu^{(i)} \in \mathbb{Z}_2^{\otimes N} \quad | \quad i \in \{0, \dots, l\}, \quad (\text{B6})$$

$$\nu^{(l)} = \nu \in \mathcal{V}, \quad (\text{B7})$$

$$\nu^{(i-1)} = \nu^{(i)} + \mathbf{u}_{a_i} \bmod 2. \quad (\text{B8})$$

Now, the procedure starts:

$$\left[\prod_{i=1}^l (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2} \right] \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k} \right] |\Theta\rangle \quad (\text{B9})$$

$$= \left[\prod_{i=1}^{l-1} (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2} \right] \frac{1}{2} [1 - (-1)^{b_l} (-1)^{\nu_{a_l}}] (-1)^{\sum_{j<a_l} \nu_j} \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k + \delta_{a_l k} \bmod 2} \right] |\Theta\rangle \quad (\text{B10})$$

$$= \left[\frac{1}{2} [1 - (-1)^{b_l} (-1)^{\nu_{a_l}^{(l)}}] (-1)^{\sum_{j<a_l} \nu_j^{(l)}} \right] \left[\prod_{i=1}^{l-1} (c_{a_i}^\dagger)^{b_i} (c_{a_i})^{1+b_i \bmod 2} \right] \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k^{(l-1)}} \right] |\Theta\rangle \quad (\text{B11})$$

$$= \left[\prod_{i=1}^l \underbrace{\frac{1}{2} [1 - (-1)^{b_i} (-1)^{\nu_{a_i}^{(i)}}]}_{\text{projector eigenvalues}} \underbrace{(-1)^{\sum_{j<a_i} \nu_j^{(i)}}}_{\text{parity signs}} \right] \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k^{(0)}} \right] |\Theta\rangle. \quad (\text{B12})$$

We again explain what has happened: first, the rightmost operator, which is either c_{a_l} or $c_{a_l}^\dagger$ depending on the parameter b_l , acts on the state according to either (B4) or (B5). We therefore combine the two relations for the absorption of this operator $(c_{a_l}^\dagger)^{b_l} (c_{a_l})^{1+b_l \bmod 2}$ in (B10). In the same fashion, all the remaining operators of the sequence are one-after-another absorbed into the state. The new state is described by the vector $\nu^{(l-1)}$ after the update. And the cycle begins anew with $(c_{a_{l-1}}^\dagger)^{b_{l-1}} (c_{a_{l-1}})^{1+b_{l-1} \bmod 2}$. From (B11) on, we use the notations (B6)–(B8) to describe partially updated occupations. By the end of this iteration, the occupation of the state is changed to $\nu^{(0)} = \nu + \mathbf{q} \bmod 2$, with the total change $\mathbf{q} = \sum_i \mathbf{u}_{a_i} \bmod 2$. Also, the coefficients of (B12) take into account sign changes from anticommutations ('parity signs' in (B12)) and the eigenvalues of the applied projections. In its entirety, (B12) denotes the resulting state, and is the main ingredient for the next step.

B.2. Parity operators and projectors

We are given the operator transform (31) and the state transform (19). We want to show that the Fermion system is adequately simulated, which means to show that the effect (B12) is replicated by (31) acting on $|\mathbf{e}(\nu)\rangle$. This is the goal of the next two steps. We start by evaluating the application of (31) on that state, up to the update operator \mathcal{U}^a . This means that the operators applied implement two things only: the parity signs of (B12), and the projection onto the correct occupational state. Note that these parity operators and projectors are applied before the update operator in (31):

$$\underbrace{\mathcal{U}^a}_{\text{update operator}} \underbrace{\left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{a_v a_w}} \right)}_{\text{parity operators}} \prod_{x=1}^l \underbrace{\frac{1}{2} \left(\mathbb{I} - \left[\prod_{y=x+1}^l (-1)^{\delta_{a_x a_y}} \right] (-1)^{b_x} \mathfrak{X}[d_{a_x}] \right)}_{\text{projectors}} \underbrace{\mathfrak{X}[p_{a_x}]}_{\text{parity operators}}. \quad (\text{B13})$$

We now commence our evaluation:

$$\mathcal{U}^a \left[\left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{a_v a_w}} \right) \prod_{x=1}^l \frac{1}{2} \left(\mathbb{I} - \left[\prod_{y=x+1}^l (-1)^{\delta_{a_x a_y}} \right] (-1)^{b_x} \mathfrak{X}[d_{a_x}] \right) \mathfrak{X}[p_{a_x}] \right] |\mathbf{e}(\nu)\rangle \quad (\text{B14})$$

$$= \mathcal{U}^a \left[\left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{avaw}} \right) \prod_{x=1}^l \frac{1}{2} \left(1 - \left[\prod_{y=x+1}^l (-1)^{\delta_{axay}} \right] (-1)^{b_x} (-1)^{d_{ax}(\mathbf{e}(\nu))} \right) (-1)^{p_{ax}(\mathbf{e}(\nu))} \right] |\mathbf{e}(\nu)\rangle \quad (\text{B15})$$

$$= \mathcal{U}^a \left[\left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{avaw}} \right) \prod_{x=1}^l \frac{1}{2} \left(1 - \left[\prod_{y=x+1}^l (-1)^{\delta_{axay}} \right] (-1)^{b_x} (-1)^{\nu_{ax}} \right) (-1)^{\sum_{j < a_x} \nu_j} \right] |\mathbf{e}(\nu)\rangle \quad (\text{B16})$$

$$= \mathcal{U}^a \left[\prod_{x=1}^l \frac{1}{2} \left(1 - (-1)^{b_x} (-1)^{\nu_{ax} + \sum_{y=x+1}^l \delta_{axay}} \right) (-1)^{\sum_{j < a_x} \nu_j + \sum_{y=x+1}^l \theta_{axay}} \right] |\mathbf{e}(\nu)\rangle \quad (\text{B17})$$

$$= \left[\prod_{x=1}^l \frac{1}{2} (1 - (-1)^{b_x} (-1)^{\nu_{ax}^{(x)}}) (-1)^{\sum_{j < a_x} \nu_j^{(x)}} \right] \mathcal{U}^a |\mathbf{e}(\nu)\rangle. \quad (\text{B18})$$

Let us describe what has happened: in (B15), the extraction property (21) is used, and we arrive at (B16) after using the property $\mathbf{d}(\mathbf{e}(\nu)) = \nu$ and the definition of the parity function. From there we go to (B17) when we merge the two products and perform rearrangements that make it easy to cast all delta and theta functions into the components of the partially updated occupations $\nu^{(i)}$, (B18).

Comparing (B18) to (B12), we notice to have successfully mimicked the same sign changes and and projections, as the coefficients in both relations match. Now it is only left to show that the state update is executed correctly. Naively, one would think that we would need to show that

$$\mathcal{U}^a |\mathbf{e}(\nu)\rangle \triangleq \left[\prod_{k=1}^N (c_k^\dagger)^{\nu_k^{(0)}} \right] |\Theta\rangle, \quad (\text{B19})$$

but this is too strong a statement. It is in fact sufficient to demand

$$\mathcal{U}^a |\mathbf{e}(\nu)\rangle = |\mathbf{e}(\nu^{(0)})\rangle = |\mathbf{e}(\nu + \mathbf{q} \bmod 2)\rangle. \quad (\text{B20})$$

For $\nu^{(0)} \in \mathcal{V}$, (B19) and (B20) is equivalent. However, it might be the case that $\nu^{(0)} \notin \mathcal{V}$, so $\nu^{(0)}$ is not encoded. This mean that (B19) is not fulfilled, since $\mathbf{d}(\mathbf{e}(\nu^{(0)})) \neq \nu^{(0)}$. It is however not necessary to include $\nu^{(0)}$ in the encoding, as for $\nu^{(0)} \notin \mathcal{V}$, the state will vanish anyways: we know from $\hat{h}_{ab} : \text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B})$, that in this case \hat{h}_{ab} must act destructively on that basis state, $\hat{h}_{ab} (\prod_k (c_k^\dagger)^{\nu_k^{(0)}}) |\Theta\rangle = 0$. This detail is implemented by the projector part of the transformed sequence (31). These projectors are, as we have just shown, working faithfully like (B12), for the transformed sequence acting on every $|\nu\rangle$ with $\nu \in \mathcal{V}$. Hence (B20) is a sufficient condition for the updated state. The proof is completed once we have verified that (B20) is satisfied with the update operator defined as in (32). This is done during the next step.

B.3. Update operator

The missing piece of the proof is to check that (32) and (33) fulfill the condition (B20). We start by verifying the condition (B20) for (33), which we have presented as special case of (32) for linear encoding functions:

$\mathbf{e}(\nu + \nu' \bmod 2) = \mathbf{e}(\nu) + \mathbf{e}(\nu') \bmod 2$. Using that property, one can in fact derive (33) from (32) directly. We now apply (33) to $|\mathbf{e}(\nu)\rangle$, but firstly we note that

$$X_j |\omega\rangle = |\omega + \mathbf{u}_j \bmod 2\rangle, \quad (\text{B21})$$

where \mathbf{u}_j is the j th unit vector of $\mathbb{Z}_2^{\otimes n}$. Using (B21) and the linearity of \mathbf{e} , we find:

$$\mathcal{U}^a |\mathbf{e}(\nu)\rangle = \left[\bigotimes_{i=1}^n (X_i)^{\sum_j A_{ij} q_j \bmod 2} \right] |\mathbf{e}(\nu)\rangle \quad (\text{B22})$$

$$= \left[\bigotimes_{i=1}^n (X_i)^{\mathbf{e}(\mathbf{q})} \right] |\mathbf{e}(\nu)\rangle \quad (\text{B23})$$

$$= |\mathbf{e}(\nu) + \mathbf{e}(\mathbf{q}) \bmod 2\rangle \quad (\text{B24})$$

$$= |\mathbf{e}(\nu + \mathbf{q} \bmod 2)\rangle, \quad (\text{B25})$$

which shows (B20) for linear encodings.

We now turn our attention to general encodings and prove the same expression for update operators as defined in (32):

$$\mathcal{U}^a |\mathbf{e}(\nu)\rangle = \left(\sum_{\mathbf{t} \in \mathbb{Z}_2^{\otimes n}} \left[\bigotimes_{i=1}^n (X_i)^{t_i} \right] \prod_{j=1}^n \frac{1}{2} (\mathbb{I} + (-1)^{t_j} \mathcal{X}[\varepsilon_j^{\mathbf{q}}]) \right) |\mathbf{e}(\nu)\rangle \quad (\text{B26})$$

$$= \left(\sum_{\mathbf{t} \in \mathbb{Z}_2^{\otimes n}} \left[\bigotimes_{i=1}^n (X_i)^{t_i} \right] \prod_{j=1}^n \frac{1}{2} (1 + (-1)^{t_j + \varepsilon_j^q(\mathbf{e}(\nu))}) \right) |\mathbf{e}(\nu)\rangle \quad (\text{B27})$$

$$= \left(\bigotimes_{i=1}^n (X_i)^{\varepsilon_i^q(\mathbf{e}(\nu))} \right) |\mathbf{e}(\nu)\rangle \quad (\text{B28})$$

$$= |\mathbf{e}(\nu) + \varepsilon^q(\mathbf{e}(\nu)) \bmod 2\rangle \quad (\text{B29})$$

$$= |\mathbf{e}(\nu) + \mathbf{e}(\nu) + \mathbf{e}(\mathbf{d}(\mathbf{e}(\nu))) + \mathbf{q} \bmod 2 \bmod 2\rangle \quad (\text{B30})$$

$$= |\mathbf{e}(\nu + \mathbf{q} \bmod 2)\rangle, \quad (\text{B31})$$

which completes the proof. We swiftly recap what has happened: in (B26), we have plugged the definition of (32) into the left-hand side of (B20). In between this equation and (B27), we have evaluated the expectation values of the extracted operators $\mathfrak{X}[\varepsilon_j^q]$. From that line to the next, the $\mathbb{Z}_2^{\otimes n}$ -sum is collapsed over the condition $\mathbf{t} = \varepsilon^q(\mathbf{e}(\nu))$. We go from (B28) to (B29) by applying (B21). Once we insert the definition (30) into (B29), it becomes obvious that the condition (B20) is fulfilled. Thus, the entire operator transform is now proven.

Appendix C. Transforming particle-number conserving Hamiltonians

In this appendix, we examine the richest symmetry to exploit for qubit savings: particle conservation. We begin by introducing the most relevant class of Hamiltonians that exhibit this symmetry, but ultimately the main goal of this appendix is to simplify the operator transform for all such Hamiltonians. Motivated by the compartmentalized recipes of the conventional mappings, (13), we suggest alternatives to the transform (31), that do not depend on the sequence length l .

Let us start by noting how easy it is to state that a Hamiltonian the total number of particles: a Hamiltonian like (11), conserves the total number of particles when every term \hat{h}_{ab} has as many creation operators as it has annihilation operators. The lengths l , implicit in the sequences \hat{h}_{ab} that occur in the Hamiltonian, are thereby determined by the field theory or model, that underlies the problem. The coefficients h_{ab} , on the other hand, are determined by the set of basis functions used. For first-principle problems in quantum chemistry and solid state physics, we usually encounter particle-number-conserving Hamiltonians with terms of weight that is at most $l = 4$:

$$H = \sum_{ij} t_{ij} c_i^\dagger c_j + \sum_{ijkl} U_{ijkl} c_i^\dagger c_j^\dagger c_k c_l, \quad (\text{C1})$$

where U_{ijkl} , t_{ij} are complex coefficients of the interaction and single particle terms, respectively. In the notation of (11), these coefficients correspond to $h_{(i,j,k,l)(1,1,0,0)}$ and $h_{(i,j)(1,0)}$. The ($l = 4$) interaction terms usually originate from either magnetism and/or the Coulomb interaction. Even for these ($l = 4$)-terms, the operator transform (31) is quite bulky, and we in general would like to have a transform that is independent of l . Before we begin to discuss such transform recipes however, we need to set up some preliminaries. First of all, we need to find a suitable code (\mathbf{e}, \mathbf{d}) , as discussed in the main part. Ideally, we would encode only the Hilbert space with the correct number of particles, M , but Hilbert spaces of other particle numbers can also be included. Assuming that the Hamiltonian visits every state with the same particle number, we must encode entire Hilbert spaces \mathcal{H}_N^m only. Secondly, we need to reorder the fermionic operators inside the Hamiltonian terms \hat{h}_{ab} . The reason for this is, that our goal can only be achieved by finding recipes for smaller sequences of constant length. In order to transform the Hamiltonian terms then, we need to invoke the anticommutation relations (2) to introduce an order in \hat{h}_{ab} , such that these small sequences appear as consecutive, distinct blocks. As we shall see, these blocks will have the shape $c_i^\dagger c_j$. So every \hat{h}_{ab} needs to be reordered, such that every even operator is a creation operator, and every odd operator an annihilator. For the ($l = 4$)-terms in (C1), this reordering means $c_i^\dagger c_j^\dagger c_k c_l \rightarrow c_i^\dagger c_l c_j^\dagger c_k - \delta_{jl} c_i^\dagger c_k$.

Let us quickly sketch the idea behind that reordering and introduce some nomenclature: instead of considering Hamiltonian terms, we realize that also the terms $c_i^\dagger c_j$ also conserve the particle number: $\mathcal{H}_N^m \rightarrow \mathcal{H}_N^m$. Let us act with $c_i^\dagger c_j$ on an encoded state. We consider a state that is not annihilated by $c_i^\dagger c_j$. Its particle number is reduced by one through c_j , but then immediately restored by c_i^\dagger . In fact, for a general sequence of that arrangement, every even operator restores the particle number in this way and every odd reduces it. We therefore call the subspace, in which we find the state after an even (odd) number of operators, the even (odd) subspace. Since all l must be even for the Hamiltonian to have particle conservation symmetry, the even subspace is the one encoded. The odd subspace, on the other hand, has one particle less, so it is $\mathcal{H}_N^{(M-1)}$, if the even one is \mathcal{H}_N^M .

C.1. Encoding the two spaces separately

In this ordering, one can find a recipe for a singular creation or annihilation operator. The strategy is to consider a second code for the odd subspace. As before (\mathbf{e}, \mathbf{d}) denotes the code for the even subspace, and now $(\mathbf{e}', \mathbf{d}')$ is encoding the odd subspace. The idea is that after an odd operator (which in this ordering is an annihilation operator), the state is updated into the odd subspace. With every even operator (which is a creation operator), the state is updated from the odd subspace back into the even one. We find:

$$c_j^\dagger \triangleq \frac{1}{2} \bar{\mathcal{U}}^{(j)} (\mathbb{I} + \mathfrak{X}[d_j]) \mathfrak{X}[p_j], \quad (\text{C2})$$

$$c_j \triangleq \frac{1}{2} \mathcal{U}^{(j)} (\mathbb{I} - \mathfrak{X}[d'_j]) \mathfrak{X}[p'_j]. \quad (\text{C3})$$

In (C3), $\mathcal{U}^{(j)}$ is defined as in (32), but its counterpart from (C2) is defined by

$$\bar{\mathcal{U}}^{(j)} = \sum_{\mathbf{t} \in \mathbb{Z}_2^{\otimes n}} \left[\bigotimes_{i=1}^n (X_i)^{t_i} \right] \prod_{i=1}^n \frac{1}{2} (\mathbb{I} + (-1)^{t_i} \mathfrak{X}[\varepsilon'_k u_j]), \quad (\text{C4})$$

with the primed functions $\varepsilon'^q, \mathbf{p}'$ defined like (30) and (29), but with $(\mathbf{e}', \mathbf{d}')$ in place of (\mathbf{e}, \mathbf{d}) .

This method relies on n qubits being feasible to simulate the odd subspace in. That is, however, not always the case. The basis set of \mathcal{H}_N^{M-1} is in general larger than \mathcal{H}_N^M , when $M > N/2$. In this way, the odd subspace can also be larger and even be infeasible to simulate with just n qubits. As a solution, one changes the ordering into odd operators being creation operators, and even ones being annihilators, like $c_k c_i^\dagger c_l c_j^\dagger$. This causes the odd subspace to become $\mathcal{H}_N^{(M+1)}$, which has a smaller basis set than \mathcal{H}_N^M . For that case (\mathbf{e}, \mathbf{d}) become the code for the odd subspace, and $(\mathbf{e}', \mathbf{d}')$ will be associated to the even subspace in (C2) and (C3).

The obvious disadvantage is that two codes have to be employed at once. However, the checksum code for instance (section 3.1.1 in the main part), comes in two different flavors already, which can be used as codes for even and odd subspaces, respectively.

C.2. Encoding the building blocks

The building blocks $c_i^\dagger c_j$ are guaranteed to conserve the particle number, so the even subspace is conserved. As a consequence, one may consider the possibility to transform the operators as the pairs we have rearranged them into. In this way, we still have a certain compartmentalization of (31). Two special cases are to be taken into account: when $i > j$, an additional minus sign has to be added, as compared to the $i < j$ case. Also, when $i = j$, all parity operators cancel and the projectors coincide. We find:

$$c_i^\dagger c_j \triangleq \begin{cases} \frac{1}{4} (-1)^{\theta_{ij}} \mathcal{U}^{(i,j)} \mathfrak{X}[p_i] \mathfrak{X}[p_j] (\mathbb{I} + \mathfrak{X}[d_i]) (\mathbb{I} - \mathfrak{X}[d_j]) & i \neq j \\ \frac{1}{2} (1 - \mathfrak{X}[d_j]) & i = j, \end{cases} \quad (\text{C5})$$

with $\mathcal{U}^{(i,j)}$ being the $l = 2$ version of (32), and \mathbf{p} and ε^q defined as usual by (29) and (30).

Appendix D. Multi-weight binary addressing codes based on dissections

With binary addressing codes, that is codes that are similar to the one presented in section 3.1.2 in the main part, even an exponential amount of qubits can be saved for systems with low particle number, but at the expense of complicated gates. For this appendix, we firstly recap the situation of section 3.1.2 and clarify what binary addressing means. Firstly, some nomenclature is introduced. We then generalize the concept of binary addressing codes to weight K codes, using results from [28]. As an example, we explicitly obtain the $K = 2$ code.

Suppose we have a system with $N = 2^r$ orbitals, and one particle in it. Our goal is to encode the basis state, where the particle is on orbital $y \in [2^r]$, as a binary number in r qubits. In this way, the state with occupational vector \mathbf{u}_y is encoded as $|\mathbf{q}^{y,r}\rangle$, with $\mathbf{q}^{y,r} \in \mathbb{Z}_2^{\otimes r}$ and $y = \text{bin}(\mathbf{q}^{y,r}) + 1$. Probing an unknown basis state, a decoding will now have components of the form

$$\omega \rightarrow \prod_{i \in [r]} (\omega_i + q_i^{y,r} + 1) \bmod 2. \quad (\text{D1})$$

Such binary functions output 1 only when $\omega = \mathbf{q}^{y,r}$. In our nomenclature, we say that in the basis state $|\mathbf{q}^{y,r}\rangle$, the particle has the coordinate y . We refer to codes that store particle coordinates in binary form, as binary addressing codes.

In the $K = 1$ case from the main part, the code words just contain the binary representation of one coordinate. The question is now how to generalize the binary addressing codes. For multi-weight codes, we have to have K sub-registers to store the addresses of K particles. Naively, one would want to store the coordinate of each particle in its respective sub-register in binary form, as we have done for $K = 1$. This however, holds a

problem. As the particles are indistinguishable, the stored coordinates would be interchangeable, the code would not be one-to-one. For the binary numbers ω^1 and ω^2 , that represent a coordinate each, this would mean $\mathbf{d}(\omega^1 \oplus \omega^2) = \mathbf{d}(\omega^2 \oplus \omega^1)$. That strategy not only complicates the operator transform, it also leads to a certain qubit overhead, as each plain word has as many code words as there are permutations of K items. Since this naive idea leaves us unconvinced, we abandon it and search for one-to-one codes instead. The key is to consider the coordinates to be in a certain format and this is where [28] comes into play. We proceed by using some relevant concepts of that paper.

Let us consider the coordinates of K particles to be given in the N -ary vector $\mathbf{x} = (x_1, \dots, x_K)$. Between those coordinates, we have imposed an ordering $x_i > x_j$ as $i > j$. Particles cannot share the same orbital, so we are excluding the cases where two coordinates are equal. Using results from [28], we transform the latter into coordinates that lack such an ordering, and where each component is an integer from a different range:

$$\mathbf{x} \rightarrow \mathbf{y} = (y_1, \dots, y_K)^\top \quad \text{with} \quad \mathbf{y} \in \bigotimes_{m=1}^K \left[\left\lceil \frac{N}{m} \right\rceil \right]. \quad (\text{D2})$$

Through that transform, each vector \mathbf{y} corresponds to a valid vector \mathbf{x} , and there is no duplication. We now represent the \mathbf{y} -coordinates by binary numbers in the code words $\omega \in \mathbb{Z}_2^{\otimes n}$, where $n = \sum_{m=1}^K \left\lceil \log \frac{N}{m} \right\rceil$:

$$\omega = \bigoplus_{m=1}^K q^{y_m \lceil \frac{N}{m} \rceil} \quad \text{with} \quad q^{ij} \in \mathbb{Z}_2^{\otimes j} \quad \text{and} \quad \text{bin}(q^{ij}) + 1 = i. \quad (\text{D3})$$

A geometric interpretation of the process portrays the vector \mathbf{x} as a set of coordinates in a K -dimensional, discrete vector space. The vectors allowed by the ordering form thereby a multi-dimensional tetrahedron. The states outside the tetrahedron do not correspond to a valid \mathbf{V} vector, so encoding each coordinate x_i in $\lceil \log N \rceil$ qubits would be redundant. We therefore dissect the tetrahedron, and rearrange it into a *brick*, as it is referred to in [28]. What is actually done is to apply symmetry operations (like point-reflections) on the vector space until the tetrahedron is deformed into the desired shape a K -dimensional, rectangular volume. The fact that the vectors to encode are now all inside a hyper-rectangle is what we wanted to achieve. We can now clip the ranges of the coordinate axes (to $\lceil \log \frac{N}{m} \rceil$) to exclude vectors the vectors outside the brick. As the values on the axes correspond to non-binary addresses, this means that the qubit space is trimmed as well, and we have eliminated all states based on not-allowed coordinates. This is where we now reconnect to our task of finding a code: the \mathbf{e} - and \mathbf{d} -functions have to take into account the reshaping process, as only the coordinates \mathbf{x} have a physical interpretation and can be decoded. The binary addresses in the code words, on the other hand, are representatives of \mathbf{y} . With with binary logic, the two coordinates have to be reconnected. We illustrate this abstract process on the example of the ($K = 2$)-code.

D.1. Weight-two binary addressing code

As an example, we present the weight-two binary addressing code on $N = 2^r$ orbitals. The integer r will determine the size of the entire qubit system $n = 2r - 1$, with two registers of size r and $r - 1$.

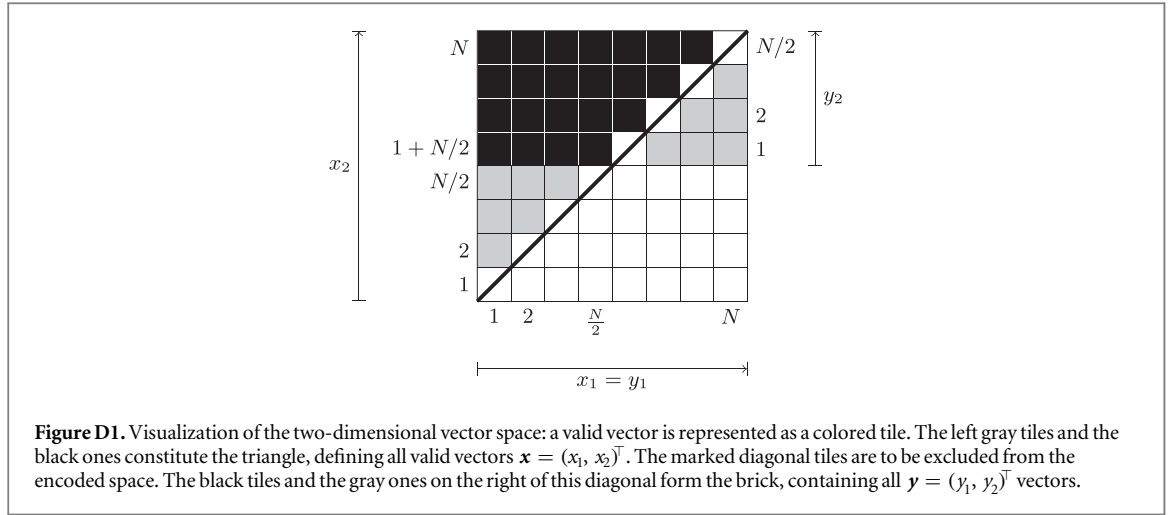
With the two registers, a binary vector $\omega = \alpha \oplus \beta$ with $\alpha \in \mathbb{Z}_2^{\otimes r}$ and $\beta \in \mathbb{Z}_2^{\otimes(r-1)}$ is defining the qubit basis. In two-dimensions, the brick turns into a rectangle and the tetrahedron into triangle. The decoding function takes binary addresses of the rectangular \mathbf{y} , and transforms them into coordinates in the triangle \mathbf{x} . The ordering condition implies hereby where to dissect the rectangle: figure D1 may serve as a visual aid, disregarding the excluded cases of $y_1 = y_2$, we find for $y_1 \in [N]$, $y_2 \in [N/2]$ and $\mathbf{x} \in [N]^{\otimes 2}$:

$$(x_1, x_2) = \begin{cases} (y_1, N/2 + y_2) & \text{for } y_1 < N/2 + y_2 \\ (y_1, N/2 - y_2 + 1) & \text{for } y_1 > N/2 + y_2. \end{cases} \quad (\text{D4})$$

This decoding is translated into a binary functions as follows: the coordinate y_1 is represented by the binary vector α and y_2 by β . For each component defined by the binary vector $\mathbf{b} \in \mathbb{Z}_2^{\otimes r}$, we have

$$\begin{aligned} d_j(\alpha \oplus \beta) = & S(\alpha, \beta) \prod_{i=1}^r (\alpha_i + q_i^{j,r} + 1) + (1 + S(\alpha, \beta))(1 + T(\alpha, \beta)) \prod_{i=1}^r (\alpha_i + q_i^{j,r}) \\ & + (1 + S(\alpha, \beta))(1 + T(\alpha, \beta)) \prod_{k=1}^{r-1} (\beta_k + q_k^{j,r}) + S(\alpha, \beta) \prod_{k=1}^{r-1} (\beta_k + q_k^{j,r} + 1) \pmod{2}, \end{aligned} \quad (\text{D5})$$

with $q^{j,r} = (q_1^{j,r}, q_2^{j,r}, \dots, q_{2^r}^{j,r})$ as defined in (D3) and we have employed two binary functions S and $T: (\mathbb{Z}_2^{\otimes r}, \mathbb{Z}_2^{\otimes(r-1)}) \rightarrow \mathbb{Z}_2$. Here, S compares the binary numbers to determine if the coordinates are left of the dissection (a black tile in figure D1)



$$S(\alpha, \beta) = \alpha_r \sum_{j=1}^{r-1} \left[\prod_{r-1 \geq i > j} (\alpha_i + \beta_i + 1) \right] (1 + \alpha_j) \beta_j + 1 + \alpha_r \pmod{2}. \quad (\text{D6})$$

The binary function T , on the other hand, is checking whether a set of coordinates is on a diagonal position (diagonally marked tiles). These excluded cases are mapped to $(0)^{\otimes r}$ altogether

$$T(\alpha, \beta) = \prod_i (\alpha_i + \beta_i) \pmod{2}. \quad (\text{D7})$$

This concludes the decoding function. Unfortunately, the amount of logic elements in the decoding will complicate the weight-two codes quite a bit, and the encoding function is hardly better. The reason for this is to find in the ordering condition: the update operations are conditional on whether we change the ordering of the coordinates represented by α and β . This is reflected in a nonlinear encoding function: we remind us that the encoding function is a map $\mathbf{e}: \mathbb{Z}_2^{\otimes 2^r} \rightarrow \mathbb{Z}_2^{\otimes (2^r-1)}$, and with $\nu \in \mathbb{Z}_2^{\otimes 2^r}$ we find

$$\begin{aligned} \mathbf{e}(\nu) = & \sum_{j=2}^{2^{r-1}} \sum_{i=1}^{j-1} (\mathbf{q}^{i,r} + \mathbf{I}^r \pmod{2}) \oplus (\mathbf{q}^{j,r-1} + \mathbf{I}^{r-1} \pmod{2}) \nu_i \nu_j \\ & + \sum_{j=2^{r-1}+1}^{2^r} \sum_{i=1}^{2^{r-1}} (\mathbf{q}^{i,r} + \mathbf{I}^r \pmod{2}) \oplus (\mathbf{q}^{j-2^{r-1},r-1}) \nu_i \nu_j \\ & + \sum_{j=2^{r-1}+2}^{2^r} \sum_{i=2^{r-1}+1}^{j-1} (\mathbf{q}^{i,r}) \oplus (\mathbf{q}^{j-2^{r-1},r-1}) \nu_i \nu_j \pmod{2}, \end{aligned}$$

with $\mathbf{q}^{i,j}$ as defined in (D3), and $\mathbf{I}^j = (1)^{\otimes j} = \mathbf{q}^{2^j j}$.

The dissecting of tetrahedrons can be generalized for codes of weight larger than two (see again [28]), but as one increases the number of dissections, the code functions are complicated even further.

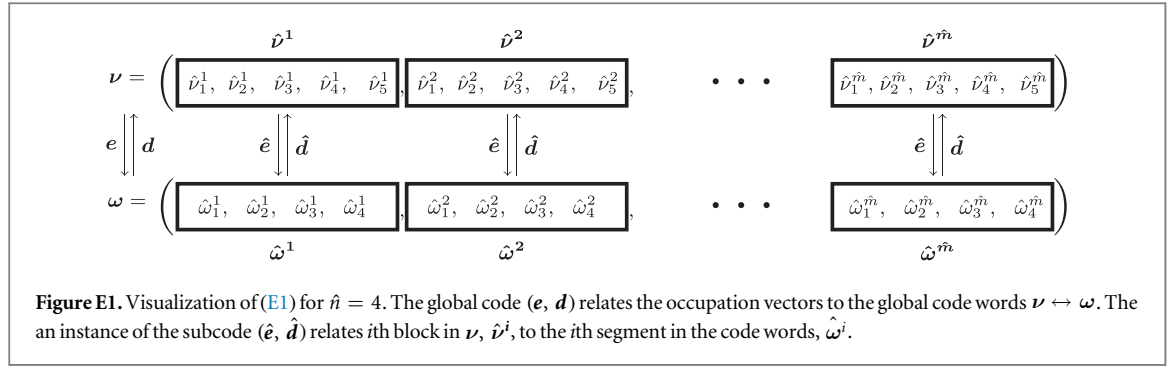
Appendix E. Segment codes

In this appendix, we provide detailed information on the segment codes. We firstly concern ourselves with the segmentation of the global code, including a derivation of the segment sizes. In another subsection we construct the segment codes themselves. The last subsection is dedicated to the adjustments one has to make to Hamiltonian, such that segment codes become feasible to use.

E.1. Segment sizes

At this point we want to sketch the idea behind the segment sizes (\hat{N}, \hat{n}) stated during section 3.1.3 in the main part, but first of all we would like to clearly set up the situation.

We consider vectors $\nu \in \mathbb{Z}_2^{\otimes N}$ to consist of \hat{m} smaller vectors $\hat{\nu}^i$ of length $\hat{n} + 1$, such that $\nu = \bigoplus_{i=1}^{\hat{m}} \hat{\nu}^i$. We call those vectors $\hat{\nu}^i$ segments of ν . The goal is now to find a code (\mathbf{e}, \mathbf{d}) to encode a basis \mathcal{V} which contains all vectors ν with Hamming weight K . For that purpose we relate the segment $\hat{\nu}^i$ to a segment of the code space, $\hat{\omega}^i$, for all $i \in [N]$. The code space segments constitute the code words in a fashion similar to the previous segmentation of ν : $\omega = \bigoplus_{i=1}^{\hat{m}} \hat{\omega}^i$. However, the length of those binary vectors $\hat{\omega}^i$ is \hat{n} , such that with $n = \hat{m}\hat{n}$ and $N = \hat{m}(\hat{n} + 1)$, the problem is reduced by \hat{m} qubits as compared to conventional transforms. We now introduce the *subcodes* $(\hat{\mathbf{e}}: \mathbb{Z}_2^{\otimes (\hat{n}+1)} \rightarrow \mathbb{Z}_2^{\otimes \hat{n}}, \hat{\mathbf{d}}: \mathbb{Z}_2^{\otimes \hat{n}} \rightarrow \mathbb{Z}_2^{\otimes (\hat{n}+1)})$, with which we encode the i th segment $\hat{\nu}^i$ as $\hat{\omega}^i$



(see figure E1). Note that we require the subcodes to inherit all the code properties. In this way we guarantee the code properties of the *global code* (\mathbf{e}, \mathbf{d}) when appending \hat{m} instances of the same subcode:

$$\mathbf{d}\left(\bigoplus_{i=1}^{\hat{m}} \hat{\boldsymbol{\omega}}^i\right) = \bigoplus_{i=1}^{\hat{m}} \hat{\mathbf{d}}(\hat{\boldsymbol{\omega}}^i), \quad \mathbf{e}\left(\bigoplus_{i=1}^{\hat{m}} \hat{\boldsymbol{\nu}}^i\right) = \bigoplus_{i=1}^{\hat{m}} \hat{\mathbf{e}}(\hat{\boldsymbol{\nu}}^i). \quad (\text{E1})$$

The orbital number being an integer multiple of the block size is of course an idealized scenario. One will probably have to add a few other components in order to compensate for dimensional mismatches.

We now set out to find the smallest segment size \hat{n} . It should be clear that \hat{n} is a function of the targeted Hamming weight K : this means K determines which segment codes are suitable for the system. The reason for this is that we need to encode all vectors with weight 0 to K inside every segment, taking into account for the up to K particles on the orbitals inside one segment. In order to include weight K vectors, the size of each segment must be at least K . If the segment size would be exactly K , on the other hand, we end up encoding the entire Fock space again. In doing so, we are not making any qubit savings. The segments must thus be larger than K . In other words, we look for an integer $\hat{n} > K$, where the sum of all combinations $\hat{\boldsymbol{\nu}} \in \mathbb{Z}_2^{\otimes(\hat{n}+1)}$ with $w_H(\hat{\boldsymbol{\nu}}) \leq K$ is smaller equal $2^{\hat{n}}$

$$2^{\hat{n}} \geq \sum_{k=0}^K \binom{\hat{n}+1}{k}. \quad (\text{E2})$$

In the case $\hat{n} = 2K$, the condition is fulfilled as identity, since exactly half of all $2^{\hat{n}+1}$ combinations are included in the sum.

E.2. Subcodes

This subsection offers a closer look at the construction of the segment subcodes $(\hat{\mathbf{e}}, \hat{\mathbf{d}})$. Let us start by considering the decoding $\hat{\mathbf{d}}$ in order to explore the nature of the binary switch $f(\hat{\boldsymbol{\omega}})$, that occurs in (38). One observes the two (affine) linear $(\mathbb{Z}_2^{\otimes \hat{n}} \rightarrow \mathbb{Z}_2^{\otimes(\hat{n}+1)})$ -maps

$$\hat{\boldsymbol{\omega}} \rightarrow \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ 0 & \dots & 0 & \end{bmatrix} \hat{\boldsymbol{\omega}} \bmod 2, \quad \hat{\boldsymbol{\omega}} \rightarrow \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ 0 & \dots & 0 & \end{bmatrix} \hat{\boldsymbol{\omega}} + \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \end{pmatrix} \bmod 2. \quad (\text{E3})$$

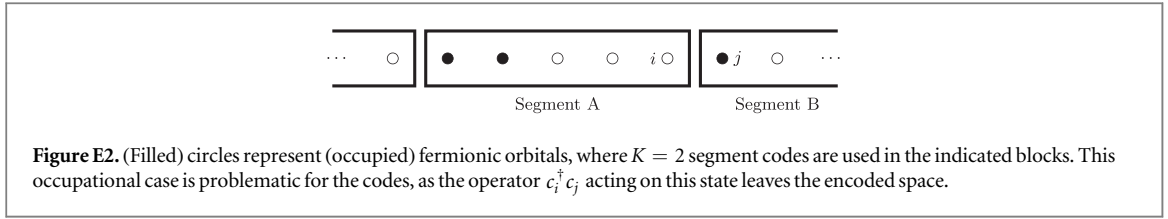
to produce together all the vectors with weight equal or smaller than K , if we input all $\hat{\boldsymbol{\omega}}$ with $w_H(\hat{\boldsymbol{\omega}}) \leq K$ into the first, and the remaining cases with $w_H(\hat{\boldsymbol{\omega}}) > K$ into the second one. Note that the last component is always zero in outputs of the first function and one in the second. Therefore, the inverse of both maps is always a linear map with the matrix $\begin{bmatrix} \mathbb{I} & | & \mathbf{I}^{\hat{n}} \end{bmatrix}$. We take this inverse as encoding (39), and the two maps (E3) are merged into the decoding (38). In order to switch between these two maps we define the binary function $f(\hat{\boldsymbol{\omega}}) : \mathbb{Z}_2^{\otimes \hat{n}} \rightarrow \mathbb{Z}_2$ such that

$$f(\hat{\boldsymbol{\omega}}) = \begin{cases} 1 & \text{for } w_H(\hat{\boldsymbol{\omega}}) > K \\ 0 & \text{otherwise.} \end{cases} \quad (\text{E4})$$

In general, one can define this binary switch in a brute-force way by

$$f(\hat{\boldsymbol{\omega}}) = \sum_{k=K+1}^{2K} \sum_{\substack{\mathbf{t} \in \mathbb{Z}_2^{\otimes 2K} \\ w_H(\mathbf{t})=k}} \prod_{m=1}^{2K} (\hat{\omega}_m + 1 + t_m) \bmod 2. \quad (\text{E5})$$

For the case $K = 1$ ($\hat{n} = 2$), the switch equals $f(\boldsymbol{\omega}) = \omega_1 \omega_2$, and for the code we recover a version of binary addressing codes, where the vector $(0, 0, 0)$ is encoded.



$$\hat{\mathbf{d}}(\hat{\omega}) = \begin{pmatrix} \hat{\omega}_1 (\hat{\omega}_2 + 1) \\ (\hat{\omega}_1 + 1) \hat{\omega}_2 \\ \hat{\omega}_1 \hat{\omega}_2 \end{pmatrix} \bmod 2, \quad \hat{\mathbf{e}}(\hat{\nu}) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \hat{\nu} \bmod 2. \quad (\text{E6})$$

In the $K = 2$ ($\hat{n} = 4$) case, this binary switch is found to

$$\text{be } f(\hat{\omega}) = \hat{\omega}_1 \hat{\omega}_2 \hat{\omega}_3 + \hat{\omega}_1 \hat{\omega}_2 \hat{\omega}_4 + \hat{\omega}_1 \hat{\omega}_3 \hat{\omega}_4 + \hat{\omega}_2 \hat{\omega}_3 \hat{\omega}_4 + \hat{\omega}_1 \hat{\omega}_2 \hat{\omega}_3 \hat{\omega}_4 \bmod 2.$$

E.3. Hamiltonian adjustments

As mentioned in section 3.1.3, in the main part, segment codes are not automatically compatible with all particle-number-conserving Hamiltonians. We show here, how certain adjustments can be made to these Hamiltonians, such that their action on the space \mathcal{H}_N^K is not changed, but segment codes become feasible to describe them with. In order to understand this issue, we begin by examining the encoded space. For that purpose we reprise the situation of (E1), where we have append \hat{m} instances of the same subcode. With segment codes, the basis \mathcal{V} contains vectors with Hamming weights from 0 to $\hat{m}K$. We have encoded all possible vectors ν with $0 \leq w_H(\nu) \leq K$, but although we have some, not all vectors with $w_H(\nu) > K$ are encoded. We can illustrate that point rather quickly: each segment has length $2K + 1$, but the subcode encodes vectors $\hat{\nu}$ with only $w_H(\hat{\nu}) \leq K$. The (global) basis \mathcal{V} is thus deprived of vectors $\nu = (\bigoplus_i \hat{\nu}^i)$ where for any segment i , $w_H(\hat{\nu}^i) > K$.

We now turn our attention to terms, which, when present in a Hamiltonian, make segment codes infeasible to use. Note, that \mathcal{V} -vectors with $w_H(\nu) \neq K$, are not corresponding to fermionic states we are interested in. In particular it is a certain subset of states with $w_H(\nu) > K$, which can lead out of the encoded space (into the states previously mentioned) when acted upon with certain fermionic operators. Let us consider the operator $c_i^\dagger c_j$ as an example, where i and j are in different segments (let us call these segments A and B). Now a basis state as depicted in figure E2, is not annihilated by $c_i^\dagger c_j$, and leads into a state with 3 particles in segment A. The problem is that the initial state is encoded in the ($K = 2$) segment codes, whereas the updated state (with the 3 particles in A) is not. In general, operators \hat{h}_{ab} , that change occupations in between segments, will cause some basis states with $w_H(\nu) > K$ to leave the encoded space. We can however adjust these terms $\hat{h}_{ab} \rightarrow \hat{h}'_{ab}$, such that $\hat{h}'_{ab}: \text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B})$, where \mathcal{B} is the basis encoded by the segment codes. We now sketch the idea behind those adjustments, before we reconsider the situation of figure E2. Note that after these adjustments have been made to all Hamiltonian terms in question, the segment codes are compatible with the new Hamiltonian. The idea is to switch those terms off for states, that already have K particles inside the segments, to which particles will be added. We have to take care to do this in a way that leaves the Hamiltonian hermitian on the level of second quantization, i.e. we have to adjust the terms \hat{h}_{ab} and \hat{h}_{ab}^\dagger into \hat{h}'_{ab} and $(\hat{h}'_{ab})'$, such that $\hat{h}'_{ab} + (\hat{h}'_{ab})'$ is hermitian. For the $K = 2$ code of figure E2, we can make the following adjustments:

$$c_i^\dagger c_j \rightarrow \left(1 - \sum_{l, k < l \in B} c_k^\dagger c_k c_l^\dagger c_l \right) c_i^\dagger c_j \left(1 - \sum_{w, v < w \in A} c_v^\dagger c_v c_w^\dagger c_w \right). \quad (\text{E7})$$

Appendix F. Conventional mappings

We now revisit the conventional transforms from section 2 in the main part, and discuss all notations that have been introduced to express it close to the appealing nomenclature of [24, 33]. In particular, we show that the relation (13) is recovered as a special case from (31) and (33). After that, we verify that such constructions satisfy the fermionic anticommutation relations. For now, however, we would like to restate the situation: a linear $n = N$ code, encoding the entire Fock space, is mediated by the quadratic matrices A and A^{-1} , such that $\mathbf{e}(\nu) = (A\nu \bmod 2)$ and $\mathbf{d}(\omega) = (A^{-1}\omega \bmod 2)$. The matrices are required to be each others inverses, so

$$\sum_{j=1}^N A_{ij} (A^{-1})_{jk} \bmod 2 = \delta_{ik}. \quad (\text{F1})$$

We now explain the form of the parity, update and flip sets. As the code is linear, the extraction operator is retrieving only Pauli strings following (22) and (24). One finds:

$$\mathfrak{X}[d_i] = \mathfrak{X}\left[\omega \rightarrow \sum_j (A^{-1})_{ij} \omega_j \bmod 2\right] = \bigotimes_{j \in [N]} (Z_j)^{(A^{-1})_{ij}} = \bigotimes_{j \in F(i)} Z_j, \quad (\text{F2})$$

$$\mathfrak{X}[p_i] = \mathfrak{X}\left[\omega \rightarrow \sum_{j < i} \sum_k (A^{-1})_{jk} \omega_k \bmod 2\right] = \bigotimes_{k \in [N]} (Z_j)^{(RA^{-1})_{ik}} = \bigotimes_{k \in P(i)} Z_k, \quad (\text{F3})$$

where $P(i)$ and $F(i)$ are the parity and flip sets with respect to i , as we defined them in section 2. The update sets $U(i)$ are obtained from update operators of linear encodings:

$$\mathcal{U}^a = \bigotimes_i (X_i)^{e_i(q)} = \bigotimes_i (X_i)^{\sum_j A_{ij} q_j} = \prod_{k \in [I]} \bigotimes_{i \in [N]} (X_i)^{A_{ik}} = \prod_{k \in [I]} \bigotimes_{i \in U(a_k)} X_i. \quad (\text{F4})$$

In order to derive (13), we would like to point out the commutation relations between Pauli strings $(\bigotimes_{u \in U(i)} X_u)$, $(\bigotimes_{v \in F(j)} Z_v)$ and $(\bigotimes_{w \in P(k)} Z_w)$. These will prove useful in verifying the fermionic commutation relations later. For commutations of update- and flip set strings we find:

$$\left(\bigotimes_{u \in U(i)} X_u\right) \left(\bigotimes_{v \in F(j)} Z_v\right) = \bigotimes_{w \in [N]} (X_w)^{A_{iw}} (Z_w)^{(A^{-1})_{wj}} = \bigotimes_{w \in [N]} (-1)^{A_{iw}(A^{-1})_{wj}} (Z_w)^{(A^{-1})_{wj}} (X_w)^{A_{iw}} \quad (\text{F5})$$

$$= (-1)^{\sum_w A_{iw}(A^{-1})_{wj}} \left(\bigotimes_{v \in F(j)} Z_v\right) \left(\bigotimes_{u \in U(i)} X_u\right) = (-1)^{\delta_{ij}} \left(\bigotimes_{v \in F(j)} Z_v\right) \left(\bigotimes_{u \in U(i)} X_u\right). \quad (\text{F6})$$

We have used the relation (F1) for the above. Similarly, for commutations of update and parity strings we have:

$$\left(\bigotimes_{u \in U(i)} X_u\right) \left(\bigotimes_{w \in P(j)} Z_w\right) = (-1)^{\theta_{ij}} \left(\bigotimes_{w \in P(j)} Z_w\right) \left(\bigotimes_{u \in U(i)} X_u\right). \quad (\text{F7})$$

Finally, we combine (F2)–(F4) with the operator from (31). Using (F5) and (F7) to move every update string $(\bigotimes_{u \in U(a_j)} X_u)$ in between the projectors and parity strings of a_j and a_{j+1} , we get

$$\mathcal{U}^a \left(\prod_{v=1}^{l-1} \prod_{w=v+1}^l (-1)^{\theta_{avaw}} \right) \left(\prod_{x=1}^l \frac{1}{2} \left(\mathbb{I} - \left[\prod_{y=x+1}^l (-1)^{\delta_{axay}} \right] (-1)^{b_x} \mathfrak{X}[d_{a_x}] \right) \mathfrak{X}[p_{a_x}] \right) \quad (\text{F8})$$

$$= \prod_{x=1}^l \left(\frac{1}{2} \left(\bigotimes_{u \in U(a_x)} X_u \right) \left(\mathbb{I} - (-1)^{b_x} \bigotimes_{v \in F(a_x)} Z_v \right) \bigotimes_{w \in P(a_x)} Z_w \right), \quad (\text{F9})$$

which is a sequence of the operators (13). The transform of a singular operator is $c_j^{(\dagger)}$ is thus derived from (31). Although we have already shown that (31) satisfies (3)–(6), but we now want to show that (13) fulfills the anticommutation relations (2) in particular. In doing so, we generally distinguish the cases $i = j$ and $i \neq j$. For $[c_j^{(\dagger)}, c_j^{(\dagger)}]_{\pm}$, we consult (F5) and find

$$c_j^{\dagger} c_j^{\dagger} = c_j c_j \triangleq \frac{1}{4} \left(\mathbb{I} - \bigotimes_{v \in F(j)} Z_v \right) \left(\mathbb{I} + \bigotimes_{w \in F(j)} Z_w \right) = 0. \quad (\text{F10})$$

We notice that for $i \neq j$, the gate transform of $c_i c_j$ ($c_i^{\dagger} c_j^{\dagger}$) properly differs by a minus sign from the transform of $c_j c_i$ ($c_j^{\dagger} c_i^{\dagger}$) due to (F7). We want to make this observation explicit for the $i \neq j$ case of $[c_i, c_j^{\dagger}]_{\pm}$:

$$c_i c_j^{\dagger} \triangleq \frac{1}{4} \left(\bigotimes_{u \in U(i)} X_u \right) \left(\mathbb{I} - \bigotimes_{v \in F(i)} Z_v \right) \left(\bigotimes_{w \in P(i)} Z_w \right) \left(\bigotimes_{u' \in U(j)} X_{u'} \right) \left(\mathbb{I} + \bigotimes_{v' \in F(j)} Z_{v'} \right) \left(\bigotimes_{w' \in P(j)} Z_{w'} \right) \quad (\text{F11})$$

$$= \frac{(-1)^{\theta_{ij}}}{4} \left(\bigotimes_{u \in U(i)} X_u \right) \left(\bigotimes_{u' \in U(j)} X_{u'} \right) \left(\mathbb{I} - \bigotimes_{v \in F(i)} Z_v \right) \left(\bigotimes_{w \in P(i)} Z_w \right) \left(\mathbb{I} + \bigotimes_{v' \in F(j)} Z_{v'} \right) \left(\bigotimes_{w' \in P(j)} Z_{w'} \right) \quad (\text{F12})$$

$$= \frac{(-1)^{\theta_{ij} + \theta_{ji}}}{4} \left(\bigotimes_{u' \in U(j)} X_{u'} \right) \left(\mathbb{I} + \bigotimes_{v' \in F(j)} Z_{v'} \right) \left(\bigotimes_{w' \in P(j)} Z_{w'} \right) \left(\bigotimes_{u \in U(i)} X_u \right) \left(\mathbb{I} - \bigotimes_{v \in F(i)} Z_v \right) \left(\bigotimes_{w \in P(i)} Z_w \right) \quad (\text{F13})$$

$$\triangleq -c_j^{\dagger} c_i. \quad (\text{F14})$$

At last, we find by explicit construction :

$$c_j^\dagger c_j \triangleq \frac{1}{2} \left(\mathbb{I} - \bigotimes_{v \in F(j)} Z_v \right), \quad c_j c_j^\dagger \triangleq \frac{1}{2} \left(\mathbb{I} + \bigotimes_{v \in F(j)} Z_v \right). \quad (\text{F15})$$

Thus, we find $[c_i, c_j^\dagger]_+ \triangleq \delta_{ij} \mathbb{I}$, and our construction (13) is in compliance with all relations in (2).

Appendix G. Notations

Symbol	Type	Informal definition
\dots		Set of integers from 1 to argument
\triangleq		Correspondence between fermionic operators/states to qubit counterparts
\mathbf{a}	$[N]^{\otimes l}$	Length- lN -ary vector describing orbitals in \hat{h}_{ab}
$A^{(-1)}$		$(N \times N)$ binary matrix defining a conventional encoding (decoding)
\mathbf{b}	\mathbb{Z}_2^l	Length- l binary vector determining operator types in \hat{h}_{ab}
\mathcal{B}		Basis of a space of Fermions on N orbitals, smaller than the Fock space
$c_j^{(\dagger)}$	$\mathcal{L}(\mathcal{F}_N)$	Fermionic annihilation (creation) operator
$(\mathbb{C}^2)^{\otimes n}$		Vector space of n -qubit states
\mathbf{d}	$\mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2^{\otimes N}$	Decoding function
\mathbf{e}	$\mathbb{Z}_2^{\otimes N} \rightarrow \mathbb{Z}_2^{\otimes n}$	Encoding function
ε^q	$\mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2^{\otimes n}$	Update function important for nonlinear encodings, see (30)
\mathcal{F}_N		Fock space restricted on N orbitals
$F(j)$	subset of $[n]$	Flip set with respect to orbital j , see (13)
\hat{h}_{ab}	$\text{span}(\mathcal{B}) \rightarrow \text{span}(\mathcal{B})$	Term in a fermionic Hamiltonian, see (11)
\mathcal{H}_N^M		Antisymmetrized Hilbert space of M indistinguishable particles on N orbitals
\mathbb{I}		Identity operator on arbitrary spaces
K	$[N]$	Targeted Hamming weight of a code
l		Length of a sequence of fermionic operators in \hat{h}_{ab}
L	$[n]$	Weight of a Pauli string
$\mathcal{L}(\dots)$		Linear operators on argument vector space
M	$[N]$	Total particle number in a system of N orbitals
n		Number of qubits
N		Number of orbitals
ν	$\mathcal{V} \subseteq \mathbb{Z}_2^{\otimes N}$	N -orbital occupation vector representing a fermionic basis state, see (18)
ω	$\mathbb{Z}_2^{\otimes n}$	Binary vector representing a product state in the n -qubit basis, see (19)
\mathbf{p}	$\mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2^{\otimes N}$	Parity function, implementing signs for parity operators
$P(j)$	subset of $[n]$	Parity set of orbital j , see (13)
\mathcal{P}		Set of single-qubit Pauli operators $\{X, Y, Z\}$
\mathbf{q}	$\mathbb{Z}_2^{\otimes N}$	Change of a vector ν by a term \hat{h}_{ab}
R		Binary $(N \times N)$ matrix, with the lower triangle (including diagonal) filled with ones
θ_{ij}	$[N]^{\otimes 2} \rightarrow \mathbb{Z}_2$	Discrete version of the Heaviside function, see (14)
\mathbf{u}_j	$\mathbb{Z}_2^{\otimes N}$ or $\mathbb{Z}_2^{\otimes n}$	Binary unit vector, just component j is one
$U(j)$	subset of $[n]$	Update set of orbital j , see (13)
\mathcal{U}^a	$\mathcal{L}((\mathbb{C}^2)^{\otimes n})$	Update operator with respect to an occupation of \mathbf{a} , see (32) and (33)
\mathcal{V}	subset of $\mathbb{Z}_2^{\otimes N}$	Defines the occupation vectors ν implementing the basis \mathcal{B} , see (18)
$w_H(\dots)$	$\mathbb{Z}_2^{\otimes N} \rightarrow [N] \cup \{0\}$	Hamming weight of a binary vector, sum of its components
\mathfrak{X}	$(\mathbb{Z}_2^{\otimes n} \rightarrow \mathbb{Z}_2) \rightarrow \mathcal{L}((\mathbb{C}^2)^{\otimes n})$	Extraction superoperator, maps binary functions into quantum gates, see (20)–(28)
\mathbb{Z}_2		Binary digits $\{0, 1\}$

References

- [1] Lloyd S 1996 Universal quantum simulators *Science* **273** 1073–8
- [2] Bravyi S B and Kitaev A Y 2002 Fermionic quantum computation *Ann. Phys., NY* **298** 210–26
- [3] Aspuru-Guzik A, Dutoi A D, Love P J and Head-Gordon M 2005 Simulated quantum computation of molecular energies *Science* **309** 1704–7
- [4] Whitfield J D, Biamonte J and Aspuru-Guzik A 2011 Simulation of electronic structure Hamiltonians using quantum computers *Mol. Phys.* **109** 735–50
- [5] Hastings M B, Wecker D, Bauer B and Troyer M 2014 Improving quantum algorithms for quantum chemistry arXiv:1403.1539
- [6] Babbush R, Love P J and Aspuru-Guzik A 2014 Adiabatic quantum simulation of quantum chemistry *Sci. Rep.* **4** 6603
- [7] Lanyon B P *et al* 2010 Towards quantum chemistry on a quantum computer *Nat. Chem.* **2** 106–11
- [8] Du J, Xu N, Peng X, Wang P, Wu S and Lu D 2010 Nmr implementation of a molecular hydrogen quantum simulation with adiabatic state preparation *Phys. Rev. Lett.* **104** 030502

- [9] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O'Brien J L 2014 A variational eigenvalue solver on a photonic quantum processor *Nat. Commun.* **5** 4213
- [10] Wang Y *et al* 2015 Quantum simulation of helium hydride cation in a solid-state spin register *ACS Nano* **9** 7769–74
- [11] O'Malley P J *et al* 2016 Scalable quantum simulation of molecular energies *Phys. Rev. X* **6** 031007
- [12] Kandala A, Mezzacapo A, Temme K, Takita M, Chow J M and Gambetta J M 2017 Hardware-efficient quantum optimizer for small molecules and quantum magnets *Nature* **549** 242
- [13] Wecker D, Bauer B, Clark B K, Hastings M B and Troyer M 2013 Can quantum chemistry be performed on a small quantum computer p 15 arXiv:1312.1695
- [14] Poulin D, Hastings M B, Wecker D, Wiebe N, Doherty A C and Troyer M 2014 The Trotter step size required for accurate quantum simulation of quantum chemistry arXiv:1406.4920
- [15] Babbush R, McClean J, Wecker D, Aspuru-Guzik A and Wiebe N 2015 Chemical basis of Trotter–Suzuki errors in quantum chemistry simulation *Phys. Rev. A* **91** 022311
- [16] Reiher M, Wiebe N, Svore K M, Wecker D and Troyer M 2017 Elucidating reaction mechanisms on quantum computers *Proc. Natl Acad. Sci.* **114** 7555–60
- [17] Shor P W 1999 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM Rev.* **41** 303–32
- [18] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502
- [19] Abrams D S and Lloyd S 1997 Simulation of many-body Fermi systems on a universal quantum computer *Phys. Rev. Lett.* **79** 2586
- [20] Wigner E P and Jordan P 1928 Über das Paulische Äquivalenzverbot *Z. Phys.* **47** 631
- [21] Somma R, Ortiz G, Gubernatis J E, Knill E and Laflamme R 2002 Simulating physical phenomena by quantum networks *Phys. Rev. A* **65** 042323
- [22] Somma R, Ortiz G, Knill E and Gubernatis J 2003 Quantum simulations of physics problems *Int. J. Quantum Inf.* **1** 189–206
- [23] Verstraete F and Cirac J I 2005 Mapping local Hamiltonians of fermions to local Hamiltonians of spins *J. Stat. Mech.* P09012
- [24] Seeley J T, Richard M J and Love P J 2012 The Bravyi–Kitaev transformation for quantum computation of electronic structure *J. Chem. Phys.* **137** 224109
- [25] Jones N C, Whitfield J D, McMahon P L, Yung M-H, Van Meter R, Aspuru-Guzik A and Yamamoto Y 2012 Faster quantum chemistry simulation on fault-tolerant quantum computers *New J. Phys.* **14** 115023
- [26] Havlíček V, Troyer M and Whitfield J D 2017 Operator locality in the quantum simulation of fermionic models *Phys. Rev. A* **95** 032332
- [27] Kivlichan I D, McClean J, Wiebe N, Gidney C, Aspuru-Guzik A, Chan G K L and Babbush R 2018 Quantum simulation of electronic structure with linear depth and connectivity *Phys. Rev. Lett.* **120** 110501
- [28] Tian C, Vaishampayan V A and Sloane N J A 2007 *IEEE Trans. Inf. Theory* **55** 1051–60
- [29] Bravyi S, Smith G and Smolin J A 2016 Trading classical and quantum computational resources *Phys. Rev. X* **6** 021043
- [30] Moll N, Fuhrer A, Staar P and Tavernelli I 2016 Optimizing qubit resources for quantum chemistry simulations in second quantization on a quantum computer *Bull. Am. Phys. Soc.* **49** 295301
- [31] Bravyi S, Gambetta J M, Mezzacapo A and Temme K 2017 Tapering off qubits to simulate fermionic Hamiltonians arXiv:1701.08213
- [32] Romero J, Olson J P and Aspuru-Guzik A 2017 Quantum autoencoders for efficient compression of quantum data *Quantum Sci. Technol.* **2** 045001
- [33] Tranter A, Sofia S, Seeley J, Kaicher M, McClean J, Babbush R, Coveney P V, Mintert F, Wilhelm F and Love P J 2015 The Bravyi–Kitaev transformation: Properties and applications *Int. J. Quantum Chem.* **115** 1431–41
- [34] Ruckenstein A E, Hirschfeld P J and Appel J 1987 Mean-field theory of high- T_c superconductivity: the superexchange mechanism *Phys. Rev. B* **36** 857
- [35] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H 1995 Elementary gates for quantum computation *Phys. Rev. A* **52** 3457
- [36] Whitfield J D, Havlíček V and Troyer M 2016 Local spin operators for fermion simulations *Phys. Rev. A* **94** 030301
- [37] Kitaev A Y 1995 Quantum measurements and the Abelian stabilizer problem arXiv:quant-ph/9511026
- [38] Cleve R, Ekert A, Macchiavello C and Mosca M 1998 Quantum algorithms revisited *Proc. R. Soc. A* **454** 339–54
- [39] Abrams D S and Lloyd S 1999 Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors *Phys. Rev. Lett.* **83** 5162
- [40] McClean J R, Romero J, Babbush R and Aspuru-Guzik A 2016 The theory of variational hybrid quantum–classical algorithms *New J. Phys.* **18** 023023
- [41] Berry D W, Childs A M, Cleve R, Kothari R and Somma R D 2015 Simulating Hamiltonian dynamics with a truncated Taylor series *Phys. Rev. Lett.* **114** 090502
- [42] Berry D W, Childs A M and Kothari R 2015 Hamiltonian simulation with nearly optimal dependence on all parameters 2015 *IEEE 56th Annual Symp. on Foundations of Computer Science (FOCS)* (Piscataway, NJ: IEEE) pp 792–809
- [43] Babbush R, Berry D W, Kivlichan I D, Wei A Y, Love P J and Aspuru-Guzik A 2016 Exponentially more precise quantum simulation of fermions in second quantization *New J. Phys.* **18** 033032
- [44] Suzuki M 1990 Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations *Phys. Lett. A* **146** 319–23
- [45] Suzuki M 1991 General theory of fractal path integrals with applications to many-body theories and statistical physics *J. Math. Phys.* **32** 400–7
- [46] Nielsen M A and Chuang I 2002 Quantum Computation and Quantum Information (College Park, MD: AAPT)